KU LEUVEN

ARENBERG DOCTORAL SCHOOL
Faculty of Engineering Science

The Design and Cryptanalysis of Post-Quantum Digital Signature Algorithms

Ward Beullens

Supervisors:
Prof. dr. ir. B. Preneel
Prof. dr. ir. F. Vercauteren

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Science (PhD): Electrical Engineering

May 2021

The Design and Cryptanalysis of Post-Quantum Digital Signature Algorithms

Ward BEULLENS

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Science (PhD): Electrical Engineering

May 2021

Preface

Don't use epigraphs as a way to make the thesis sound more impressive; that's a waste of time because nobody reads a thesis.

– Nate Eldredge, `academia.stackexchange.com`

Time flies when you are having fun, and so three and a half years of PhD research have flown by as if they started only yesterday. In these years, I had the fortune of learning lots of interesting things, visiting faraway places, and meeting many wonderful people. I feel very privileged for being able to spend my time working on fun problems, reading interesting papers, and collaborating with wonderful colleagues (and even be paid to do so!). I have many people to thank for making this possible and enjoyable.

First of all, I want to thank my supervisors Bart and Fre. You have always been there to give me advice, answer my questions, proofread my papers, and otherwise make time for me when I needed it. Also, thank you for giving me the freedom to work on the topics that sparked my interest and for generously supporting my travels to many conferences.

I also want to thank my co-authors, Alan, Bart, Simon, Fre, Thorsten, Hoeteck, Cyprien, Tim, Aleksei, Guiseppe, Shu, Federico, Robi, Lucas, Jean-Charles, Eliane, Gilles, Jacques, Ludovic (in no particular order), without whom the research in this thesis would not have been possible. It was a pleasure to work with you. (Most of the time anyway.) I look forward to working with you again in the future. In particular, I want to thank Alan for supervising my master thesis project on multivariate cryptography. If it weren't for that project I would most likely not have started a career in cryptography.

I want to thank all the members of COSIC for making it such a nice place to work. I thoroughly enjoyed the board game nights, the coding competitions,

going to Rock Werchter, and even the seminars, public-key, and coed meetings. A big thanks to the supporting staff for their invaluable help, and in particular Péla, for booking my PhD defense on the correct date despite my best efforts of sabotaging you.

I am grateful to the FWO (Fonds Wetenschappelijk Onderzoek) for supporting my research with an SB fellowship.

Last but not least, I want to thank my family and friends. My parents, for giving me so many opportunities in life and for encouraging me to do the things I like doing. My friends, for providing me with interesting office decorations which continue to raise eyebrows. Matthew, for your relentless encouragements whenever I needed them, and for helping me keep up the appearance of a work-life balance.

Abstract

Quantum computers will be able to break all currently deployed public-key cryptographic algorithms. As these algorithms are crucial to secure our IT infrastructure, we urgently need to transition to so-called post-quantum cryptographic algorithms, which can resist attacks from attackers with access to large-scale quantum computers. There already exist several candidate quantum-resistant algorithms, but despite substantial progress in the last decade, they are unfortunately still less efficient and less understood than the algorithms that are currently deployed, especially in the case of digital signature algorithms, which are the focus of this thesis. This makes the necessary transition to post-quantum algorithms more difficult.

To help solve this problem, this thesis makes contributions to three objectives: to diversify the set of available signature schemes, to investigate the (in)security of supposedly post-quantum signatures through cryptanalysis, and to design digital signature algorithms that are more efficient in terms of speed, key size, and signature size.

Diversifying post-quantum signature schemes. There is a sizeable collection of mathematical building blocks that we can build post-quantum cryptography with, for example, there are algorithms based on lattices, error-correcting codes, hash functions, isogenies, and algorithms based on systems of multivariate quadratic equations. However, researchers have been struggling to produce truly practical signature schemes with these building blocks. (i.e., schemes that are as fast and as compact as the pre-quantum algorithms we use today). Therefore, it is important to keep searching for other areas of mathematics that we can use to build cryptography from. In this thesis we introduce the most common mathematical foundations for post-quantum signatures. Then, we introduce signature schemes based on different mathematical structures: we designed PKP-DSS and SUSHSYFISH, two signature schemes based on permuted kernels, and LegRoast, a signature scheme based on the Legendre symbol. Performance-wise, the new schemes are not yet as good as the best post-quantum signature schemes (the main issue is their signature size, which is around 10 KB, roughly a factor 10 larger than the most compact lattice-based signature schemes.). However, independent researchers have already made significant performance improvements on the PKP-DSS scheme, so there is hope that these schemes will someday be competitive with state-of-the-art post-quantum signature schemes (which have been optimized extensively during the last decade).

Cryptanalysis. Unfortunately, we cannot unconditionally prove that a signature scheme is secure. Therefore, to gain confidence in the security of an algorithm, we have to resort to analysing it and looking for weaknesses. This is called cryptanalysis. If after extensive study no weaknesses are found, then the algorithm could be secure. In this thesis, we define what it means for a signature scheme to be (in)secure, and then we report on our cryptanalytic work. We found weaknesses in multiple signature schemes that were proposed as candidates for standardisation. The most important result was the discovery of weaknesses in Rainbow, one of the finalists of the NIST standardization project. Due to this discovery, the selection of Rainbow for standardization seems unlikely. As such, the research presented in this thesis might have prevented a weak algorithm from becoming standardized and deployed at a large scale.

Designing more efficient signature schemes. Finally, we give a brief introduction to how digital signature schemes are designed, before moving on to some new signature schemes I designed myself. These new algorithms are more efficient than other post-quantum signatures which are based on the same mathematical building blocks. For example, one of the new schemes is CSI-FiSh, a signature scheme based on isogenies between supersingular elliptic curves. CSI-FiSh is more than a factor 300 faster and has signatures that are a factor 3 smaller than other signatures based on the same mathematical foundation.

Beknopte samenvatting

Quantumcomputers zullen in staat zijn om alle publieke-sleutel cryptografische algoritmes die momenteel in gebruik zijn te breken. Om onze IT-infrastructuur veilig te houden, moeten we daarom dringend overstappen op zogenaamde post-kwantum cryptografische algoritmes, die bestand zijn tegen aanvallers met toegang tot grootschalige kwantumcomputers. Er bestaan al verschillende kandidaat-kwantumbestendige algoritmes, maar ondanks veel vooruitgang in het afgelopen decennium zijn ze helaas nog steeds minder efficiënt en minder goed begrepen dan de algoritmes die momenteel in gebruik zijn, vooral in het geval van digitale handtekeningsalgoritmes, die het onderwerp van dit proefschrift zijn. Dit bemoeilijkt de noodzakelijke overgang naar post-kwantum algoritmes.

Om dit probleem op te lossen, levert dit proefschrift bijdragen aan drie doelstellingen: het diversifiëren van de reeks beschikbare handtekeningsalgoritmes, het onderzoeken van de (on)veiligheid van bestaande post-kwantum handtekeningen door middel van cryptanalyse, en het ontwerpen van nieuwe digitale handtekeningsalgoritmes die efficiënter zijn in termen van snelheid, sleutelgrootte en handtekeninggrootte.

Het diversifiëren van post-kwantum handtekeningsalgoritmes. Er is al een aantal wiskundige bouwstenen waarmee we post-kwantumcryptografie kunnen bouwen. Er zijn bijvoorbeeld algoritmen op basis van roosters, foutverbeterende codes, hashfuncties, isogenieën en algoritmes op basis van systemen van multivariate kwadratische vergelijkingen. Onderzoekers hebben echter moeite om met deze bouwstenen echt praktische handtekeningsalgoritmes te produceren. (d.w.z. schema's die net zo snel en compact zijn als de pre-kwantum algoritmes die we momenteel gebruiken). Daarom is het belangrijk om te blijven zoeken naar andere gebieden van de wiskunde waarop we cryptografie kunnen bouwen. In dit proefstuk introduceren we eerst de meest voorkomende wiskundige bouwstenen voor post-kwantum handtekeningsalgoritmes. Vervolgens introduceren we handtekeningsalgoritmes op basis van nieuwe wiskundige structuren: we ontwierpen PKP-DSS en

SUSHSYFISH, twee handtekeningsalgoritmes gebaseerd op gepermuteerde kernen, en LegRoast, een algoritme gebaseerd op het Legendre-symbool. Qua prestaties zijn de nieuwe schema's nog niet zo goed als de beste post-quantum handtekeningsalgoritmes (het belangrijkste probleem is de grootte van de handtekening, die met ongeveer 10 KB ongeveer een factor 10 groter dan de meest compacte op roosters gebaseerde algoritmes.). Onafhankelijke onderzoekers hebben het PKP-DSS algoritme echter al aanzienlijk kunnen verbeteren, dus er is hoop dat deze algoritmes ooit kunnen concurreren met de beste post-kwantum algoritmes (die het afgelopen decennium uitvoerig zijn geoptimaliseerd).

Cryptanalyse. Helaas kunnen we niet onvoorwaardelijk bewijzen dat een handtekeningsalgoritme veilig is. Om vertrouwen te krijgen in de veiligheid van een algoritme, moeten we daarom onze toevlucht nemen tot het analyseren en zoeken naar zwakke punten. Dit heet cryptanalyse. Als er na uitgebreid onderzoek geen zwakke punten worden gevonden, kan het algoritme veilig zijn. In dit proefschrift, definiëren we wat het betekent voor een handtekeningsalgoritme om (on)veilig te zijn, en vervolgens rapporteren we over mijn cryptanalytische werk. Ik vond zwakke punten in meerdere algoritmes die werden voorgedragen als kandidaat om gestandaardiseerd te worden. Het belangrijkste resultaat was de ontdekking van zwakke punten bij Rainbow, een van de finalisten van het NIST post-kwantum standaardisatieproject. Door deze ontdekking lijkt de selectie van Rainbow voor standaardisatie onwaarschijnlijk. Op die manier heeft mijn onderzoek, waarschijnlijk kunnen voorkomen dat een zwak algoritme gestandaardiseerd werd en op grote schaal in gebruik genomen werd.

Het ontwerpen van efficiëntere handtekeningsalgoritmes. Ten slotte geven we een korte inleiding over hoe digitale handtekeningsalgoritmes worden ontworpen. Daarna gaan we verder met enkele nieuwe handtekeningsalgoritmes die ik zelf heb ontworpen. Deze nieuwe algoritmes zijn efficiënter dan andere post-kwantum algoritmes die op dezelfde wiskundige bouwstenen zijn gebaseerd. Een van de nieuwe schema's is bijvoorbeeld CSI-FiSh, een handtekeningschema gebaseerd op isogenieën tussen supersinguliere elliptische krommes. CSI-FiSh is meer dan een factor 300 sneller en heeft handtekeningen die een factor 3 kleiner zijn dan andere handtekeningen die op dezelfde wiskunde zijn gebaseerd.

List of Abbreviations

CSI-FiSh Commutative Supersingular Isogenly Fiat-Shamir. 54

ECC Elliptic Curve Cryptography. 7

ECDSA Elliptic Curve Digital Signature Algorithm. 8

EUF-CMA Existential unforgeability under chosen message attacks. 37, 38, 41, 43, 47, 51, 55, 56, 60

FDH Full Domain Hash. 47

FIPS Federal Information Processing Standards. 12

KB kilobyte, 1024 bytes. 28

LWE Learning With Errors. 28

MB megabyte, 1024 kilobytes. 34

MPC Multi-Party Computation. 32

MQ Multivariate Quadratic. 18–20

MUDFISH Multivariate quadratic Fiat-Shamir. 55

NIST US National Institute of Standards and Technology. 12

PKP Permuted Kernel Problem. 30

PQC Post-Quantum Cryptography. 12

PRF Pseudo-Random Function. 31

QROM Quantum Random Oracle Model. 56

RSA Public-key cryptosystem due to Rivest, Shamir, and Adleman. 6

SIS Short Integer Solution. 28

SP NIST Special Publications. 12

SUF-CMA Strong unforgeability under chosen message attacks. 38

SUSHSYFISH Shuffled solution to homogeneous linear system Fiat-Shamir. 55

SVP Shortest Vector Problem. 27, 28

UOV Unbalanced Oil and Vinegar. 42

UUF-KOA Universal unforgeability under key-only attacks. 36, 37, 40, 41, 43, 47, 52

Contents

List of Figures

List of Tables

Part I

Design and Cryptanalysis of Post-Quantum digital signatures

Chapter 1

Introduction

NET::ERR_CERT_AUTHORITY_INVALID
Your connection isn't private. Attackers might be trying to steal your passwords, messages, or credit cards.

— Google Chrome 87.0.4280.88

We live in an increasingly digital world, where we use electronics to communicate with each other, make purchases, and access our fridges, toasters, and security cameras. This world would not be possible without cryptographic techniques to secure our networks and applications. End users might not be aware of the importance of cryptography, or even think that cryptography is a nuisance because cryptography is only noticeable when something goes wrong (e.g., when your browser doesn't show you a webpage because its certificate expired). However, in a world without cryptography, eavesdroppers could listen in on your private communications, criminals could plunder your bank account and terrorists could remotely take control of critical infrastructure. Alarmingly, this hypothetical world could soon become a reality, because the vast majority of our cryptographic systems rely on techniques that will become insecure when the first sufficiently large quantum computers are built.

1.1 Digital signature algorithms

Historically, if two persons wanted to communicate securely, they needed to agree on some secret way of encrypting and decrypting messages. For example, they could agree to replace each occurrence of the letter ‘a’ by the letter ‘p’, each letter ‘b’ by the letter ‘c’, and so on. A more advanced example is Enigma encryption, which was used by the German military forces during World War II. Each day, the Enigma machine needed to be reconfigured according to the secret configuration of the day. If a message encrypted with one configuration was decrypted with an Enigma machine in a different configuration, the message would be completely unintelligible.

In modern-day cryptography, this kind of encryption is referred to as symmetric-key encryption, because both parties need to know the same secret information (known as the *secret key*) to encrypt and decrypt messages. Nowadays symmetric-key cryptography is very efficient and widely used, with the most popular algorithm being the Advanced Encryption Standard (AES). If you want to use symmetric key-cryptography there is a practical problem though: How do you securely agree on a secret key? During World War II, the daily keys were distributed in codebooks, which were high-value targets for allied forces. If the allies managed to recover one book, they could listen in on the communications of an entire network, for as long as keys from that book were used.

Public-key encryption. Luckily, we don’t all have our water-soluble codebooks which we are instructed to flush down the toilet when an intruder enters our house. Where then do we get our secret keys to send WhatsApp messages to our friends and buy products online? A solution to this problem was proposed in the visionary paper of Diffie and Hellman [27], who proposed a new kind of cryptography called *public-key cryptography.* In this kind of cryptography a user can generate a pair of keys, a secret key to keep to himself, and a public key which he can distribute freely. Any person can use the public key to encrypt a message, but only the person with the corresponding secret key can decrypt and read the message. This approach solves the problem of distributing secret keys because each person can generate their own secret key, and only public keys (which are not sensitive) have to be distributed. The downside of this approach is that public-key algorithms are computationally demanding, which makes it impractical to encrypt large amounts of data such as video files. To solve this problem, hybrid encryption is often used: if person A wants to send a large amount of data to person B, he first uses a public key algorithm to securely send a secret key to person B. In the next step, they use the shared secret key to communicate large amounts of data with an efficient symmetric encryption scheme.

Public-key signatures. Encryption algorithms can keep your messages confidential, but they do not guarantee that the entity you are communicating with is authentic. To solve this problem, we need a different cryptographic technique, which is the main topic of this thesis: Digital Signature Algorithms. Much like handwritten signatures, a digital signature can be appended to a message to convince the reader that the message is authentic. A digital signature scheme consists of 3 algorithms:

- The first is a randomised **key generation** algorithm, that produces a pair of keys. A secret key which the signer keeps to himself, and a corresponding public key, which he can advertise publicly.
- The second is a **signing** algorithm, that takes a message and a secret key as input and produces a digital signature that can be appended to the message.
- The last is a **verification** algorithm, that takes a message, a public key, and a digital signature as input and outputs accept or reject signaling whether the signature is deemed valid or not.

Digital signature algorithm are designed in such a way that it is impossible to create a valid signature without the knowledge of the secret key. Therefore, digital signatures can guarantee the authenticity of messages, assuming you can guarantee the authenticity of the public key and that you can keep the secret key secret. If the receiver of a signed message can successfully verify the signature for an authentic public key, then the authenticity of the message is guaranteed, because no one else could have produced the signature. A formal description of some security properties of digital signature algorithms will be given in Chapter 3.

Applications of digital signatures. Since the invention of the first digital signature scheme in 1977, digital signatures have become an indispensable tool for computer security. Digital signatures are commonly used to achieve 3 security goals:

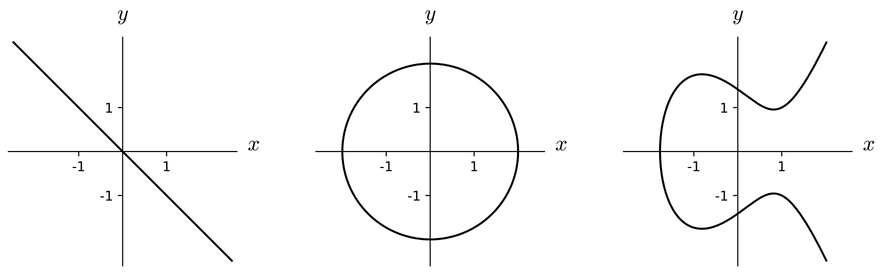
- **Entity Authentication.** As mentioned before, digital signatures can be used to authenticate entities. An important example is the TLS protocol, which is used to secure most of the traffic on the Internet. When a user browses to `example.com` he gets connected to a server on the internet. It could be that the server is really controlled by `example.com`, but it could also be a fraudulent server that tries to impersonate `example.com`. In the TLS protocol, the user sends a random message to the server, and the server responds with a digital signature for that message. The user then verifies if the signature is valid under the public key of `example.com`. If this is the case, then the user can be sure that he is talking to a legitimate server, because only servers controlled by `example.com` have the secret key capable of producing valid signatures[1].

- **Data Authentication.** Digital signatures can also ensure the authenticity of data, meaning that the sender is authentic and that the data has not been altered in transit. This is so, because altering a message invalidates the digital signature. This is important for the practice of code-signing: typically, if you download an app for your smartphone or if your operating system receives a software update, the software is signed by the author of the software (or the app store). If the signature is not valid, the software will not be installed. This prevents attackers from inserting malware in legitimate software[2].
- **Non-repudiation of origin.** Once the holder of the secret key has signed a message, he can not dispute the authorship of the message at a later point because no other person is capable of producing the valid signature. This is why digital signatures can be used to electronically sign legal documents. In many countries, including EU countries, electronic signatures have legal significance similar to handwritten signatures. For the same reason, digital signatures are used in modern credit card payments: when a purchase is made, the EMV chip on the credit card produces a digital signature for the transaction. This authenticates the credit card, but it also prevents the holder of the credit card from denying having approved of the purchase.

RSA signatures. The concept of digital signatures was introduced by Diffie and Hellman in their 1976 paper [27], but the first concrete digital signature scheme was only proposed a year later by Rivest, Shamir, and Adleman [60]. A secret key for the RSA cryptosystem consists of two large randomly chosen primes p and q, while the public key is their product $N = pq$ (along with some exponent e). If an attacker can factorise the integer N to find its two prime factors p and q, then he can recover the secret key from the public key, which means he can sign arbitrary messages as if he was an honest signer. Therefore, the RSA cryptosystem can only be secure if the problem of factoring large integers is hard. When the RSA algorithm was first proposed, the authors estimated that it would take a computer 3.8 billion years to factor a 200 digit-long number (i.e., 663 binary digits), which is almost as long as the current age of planet Earth. However, due to algorithmic advances and advances in computer technology the first 200 digit RSA modulus was factored in 2004 [6]. At the

[1]The randomness is necessary to prevent a replay attack. If the same message was signed for each connection, then the attacker could browse to `example.com` to get a valid signature, and then use that valid signature to impersonate `example.com`.

[2]Of course, this does not help if the attacker manages to insert the malware before the author signs his code, as happened in the recent SolarWinds attack which breached many branches of the US government and companies such as Microsoft, Cisco, and Equifax.



The line $x + y = 0$ The circle $x^2 + y^2 = 4$ The elliptic curve $y^2 = x^3 - 2x + 2$

Figure 1.1: Three geometric objects defined as the solution sets of an equations in the variables x and y.

time of writing, the largest RSA modulus known to be factored is 250 digits long and was factored with 2700 core-years of computing effort [13]. Nowadays it is recommended to use RSA with a 617 digit modulus (2048 bits) to resist classical attackers. RSA signatures have long been the most widely used digital signature algorithm, but in the last decade, digital signature algorithms based on elliptic curves have started to gain market share because they are more efficient and because they have smaller key and signature sizes.

Elliptic curve signatures. Elliptic curve cryptography (ECC) is a family of cryptographic algorithms based on geometric objects called *elliptic curves*. Much like a line consists of the points (x, y) in the plane satisfying $x = a + by$, and a circle consists of the points satisfying $(x - a)^2 + (y - b)^2 = r^2$, an elliptic curve can be thought of as the set of points satisfying an equation

$$y^2 = x^3 + ax + b\,,$$

together with a point at infinity, which we denote by O. If the values of a and b are integers modulo a prime p, the elliptic curves is said to be defined modulo p. Similar to how one can add points on the number line, 2 points on an elliptic curve can also be added together to form a new point on the curve[3]. This addition law turns the points of the elliptic curve (including O) into a group with neutral element O. This means that for each point P we have $P + O = P$, and for each point P there exists a point Q such that $P + Q = O$ (and we

[3]The addition law is not as simple as adding numbers, but there are concrete formulas to compute the sum $(x_1, y_1) + (x_2, y_2)$ in terms of x_1, y_1, x_2 and y_2.

denote this point Q by $-P$). For a point P on an elliptic curve E, we define

$$[k]P = \underbrace{P + \cdots + P}_{\text{sum of } k \text{ copies of } P} .$$

Given the point P and the integer k one can efficiently compute $[k]P$, but if you are given two points P and Q on an elliptic curve mod p, it could be very hard to find a value for k such that $Q = [k]P$. This problem is called the discrete logarithm problem on elliptic curves. For certain curves, this problem can be very hard to solve computationally. In the worst case, classical algorithms for solving the discrete logarithm problem take on the order of $\sqrt{p}$ steps to find k, where p is the prime modulo which the curve is defined. Therefore, if you use a well-chosen curve modulo a 256-bit prime p, then solving the problem takes roughly 2^{128} steps, which is prohibitively expensive. There exist multiple digital signature algorithms that rely on the hardness of the discrete logarithm problem for their security, such as ECDSA signatures [1] and Schnorr signatures [63]. These algorithms use a fixed base point P on a curve E modulo a certain prime p, chosen such that the discrete logarithm problem is hard. The public key then consists of a point Q on the elliptic curve, while the secret key is the discrete logarithm of Q relative to P, i.e., the secret key is the value k such that $[k]P = Q$. If the attacker can solve the discrete logarithm problem, then he can recover the secret key from the public key and use it to sign arbitrary messages. Therefore, these elliptic curve digital signature algorithms are only secure if the attackers are not able to solve the discrete logarithm problem.

Unfortunately, there do exist very efficient *quantum* algorithms for factoring integers and finding discrete logarithms, which means that if an attacker has a quantum computer that is powerful enough to run these algorithms, the RSA and ECC signature algorithms are no longer secure.

1.2 Quantum computing

Quantum computing is an upcoming technology that aims to exploit quantum phenomena, such as superposition and entanglement, to perform certain computational tasks much more efficiently than possible with traditional computing. Unlike traditional computers, which are at any point in time in some well-defined state, quantum computers can be in exponentially many states at the same time (more precisely, the quantum computer is in a superposition of all the states). For example, register A in a classical 64-bit computer could hold an integer x in the range from 0 to $2^{64} - 1$. If you call the squaring function, the computer will compute x^2 and store it in register B. On a quantum computer, register A could hold *all* the numbers from 0 to $2^{64} - 1$ *at the same time*. If you then call the function, the quantum computer will compute the squares of all these 2^{64} numbers and store them simultaneously in register B. In this way, a quantum computer can simultaneously do $2^{64} = 18\,446\,744\,073\,709\,551\,616$ computations! This is why quantum computers can be much more powerful than traditional computers. However, the laws of quantum mechanics prevent us from looking at all the squares in register B. If one was to peek at register B, the state of register B collapses to a classical state, containing only a single square. At this point, the reader might think it is useless to do 2^{64} computations if you can only learn one of the results. And indeed, for a long time it was not clear if quantum computers would be useful or not (regardless of whether they can be built).

It wasn't until the early '90s that computer scientists were able to find tasks that quantum computers can do much more efficiently than classical computers [26, 67]. Initially, these tasks were artificial and designed to be easily solvable by quantum computers, but in 1994, Shor designed a quantum algorithm that outperforms classical computers on two tasks that can actually be considered useful: finding the prime factors of large integers and computing discrete logarithms in cyclic groups [65]. It is incredibly unfortunate that these two tasks are exactly the two problems that public-key cryptography is built upon. As discussed earlier, the most widely used public-key algorithms are RSA, which can be broken if the attacker can factor large integers, as well as the DH and ECDSA algorithms, which can be broken if the attacker can compute discrete logarithms. This means that most cryptographic systems will be vulnerable to attackers with access to a quantum computer that is sufficiently powerful to run Shor's algorithm.

Since the last few years, the effort towards building useful quantum computers has heated up and shifted from academia to industry. Major players such as IBM, Intel, Microsoft, Google, and various startups are competing and regularly breaking each other's records, progressing towards more and more powerful quantum computing devices. The first useful quantum computers will likely be too noisy to break cryptography, but they would still be useful for simulating the behavior of molecules, which could result in more efficient catalysts, drugs, or batteries. These applications have the potential to generate billions in value and propel the field of quantum computing further.

In contrast to the early applications, it (currently) seems that fault-tolerant quantum computing is required to break cryptography. This can be achieved with so-called quantum error-correcting codes, which measure and correct small errors that inevitably occur during longer computations. This causes some overhead which makes fault-tolerant quantum computing more expensive than noisy quantum computing. Still, a majority of quantum computing experts estimate that there is a probability of 50% or more that quantum computing

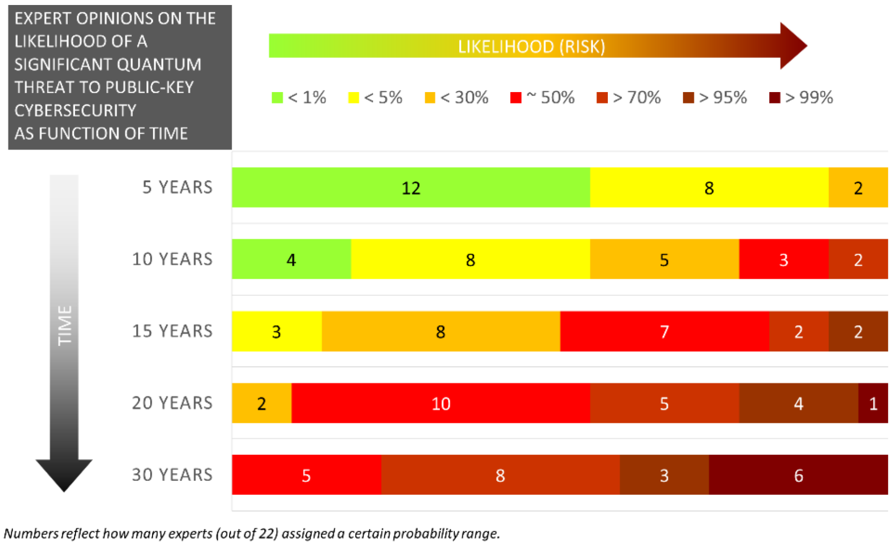


Figure 1.2: Results from a survey among 22 quantum computing experts about the treat of quantum computing on cryptography, from the quantum threat timeline report [52]

poses a significant risk to public-key cryptography in 13 years (2034), according to a 2019 survey [52] (see Figure 1.2).

1.3 Post-quantum cryptography

Luckily, there is no fundamental reason why cryptography should be vulnerable to quantum computers, and indeed some parts of cryptography, including most of symmetric-key cryptography, are largely unaffected by the advances in quantum computing. It just so happened that through a stroke of bad luck, most of public-key cryptography was built on the hardness of factoring integers and solving discrete logarithms, which turned out to be two of the relatively few tasks that quantum computers excel at. A common misconception is that we need quantum computers to secure our communication from attackers with quantum computers. This is not the case. There already exist a variety of new cryptographic algorithms, designed to be secure against quantum adversaries, and which we can run on the classical computers we have today. This is known as post-quantum cryptography (not to be confused with quantum cryptography, which is cryptography designed to exploit phenomena from quantum physics such as entanglement).

There are many branches of post-quantum cryptography, each based on the hardness of a different mathematical problem. The most prominent branches are:

- **code-based cryptography**, which relies on the hardness of problems related to error-correcting codes, such as syndrome decoding in random linear codes,
- **hash-based cryptography**, which relies on the problem of finding collisions or preimages for cryptographic hash functions,
- **lattice-based cryptography**, which relies on problems related to lattices, such as finding the point of a high-dimensional lattice that is closest to a given point outside the lattice,
- **multivariate quadratic cryptography**, which relies on the hardness of finding a solution to a system of multivariate quadratic equations, and
- **isogeny-based cryptography**, which relies on the hardness of problems involving isogenies between elliptic curves.

Chapter 2 deals with the various hardness assumptions underlying these branches in more detail.

Research in many of these branches predates Shor's algorithm and the threat of quantum computing. For example, the code-based McEliece cryptosystem [50] dates back to 1978 and is almost as old as the RSA system. However, because none of these branches had so-far produced any algorithms that could compete with RSA, DH, and Elliptic curve algorithms, they had mostly remained in the background of cryptography. One exception is lattice-based cryptography, which had been pursued since 1996 [2] for its strong theoretical security guarantees, and later because it proved to be a versatile tool for constructing advanced primitives such as fully homomorphic encryption [39].

The threat of quantum computing has reinvigorated research efforts in all the areas of post-quantum cryptography, in particular towards making the cryptosystems practical by optimising the algorithms, proposing concrete parameter sets, and writing efficient and side-channel secure implementations. Despite substantial progress, post-quantum cryptography is still not as practical or mature as the time-tested cryptographic techniques that are in use today. Overall, post-quantum algorithms are slower or need more bandwidth and storage (e.g., due to large keys) than the pre-quantum alternatives they are meant to replace. This, in combination with the fact that few post-quantum algorithms have been standardised, can discourage cryptography users to make the transition to post-quantum cryptography.

It might seem a good idea to wait for post-quantum cryptography to mature further before making the transition to post-quantum cryptography. Nevertheless, in certain situations, there are strong arguments for making the transition sooner rather than later. One example is secure web browsing where the adoption of new algorithms is a slow process that could take up to 15 years. The sooner we start the transition to post-quantum cryptography, the larger the share of post-quantum secure connections will be on the day that sufficiently powerful quantum computers become available. But the applications that most urgently need post-quantum cryptography are perhaps those that need to protect data that will remain valuable and sensitive for a long period of time, such as personal health records. This is because there is a retroactive risk: attackers can collect encrypted communication today, and store it patiently for 15 to 25 years until quantum computers can recover the sensitive information.

NIST PQC project. In 2016, motivated by progress in the development of quantum computers, including theoretical techniques for quantum error correction, the US National Institute of Standards and Technology (NIST) decided that it would be prudent to begin developing standards for post-quantum cryptography. To this end, NIST defined five security levels and launched a call for proposals [55] to invite the public to submit public key algorithms that achieve one or more of these security levels. The call resulted in 82 submissions, of which 69 were allowed to proceed to the first round of the competition-like process. NIST solicited comments from the cryptographic community and monitored the lively discussions on the online NIST PQC forum [54], as a part of its evaluation process, which spanned multiple rounds. A total of 26 algorithms proceeded to the second round of the process [3], while only 7 finalists proceeded to the third round [4][4]. NIST aims to select a subset of the 7 finalists and standardise them as Federal Information Processing Standards (FIPSs) or Special Publications (SPs).

In the past, NIST followed a similar process to standardise block ciphers and cryptographic hash functions. This resulted in the AES and SHA-3 standards, which are now very widely deployed and have been tremendously successful at helping to establish trust in IT systems around the world. NIST estimates the benefits of AES in the period 1996-2017 on the US economy alone at 250 billion US dollars [47]. The post-quantum project is different from the AES and SHA-3 competitions because the science of post-quantum cryptography

[4]In addition to the 7 finalists, 8 alternate algorithms with potential for future standardisation were selected.

is still in its infancy and there is much uncertainty about the security of the new algorithms. This is exemplified by the many weaknesses that have been discovered in the submitted algorithms. NIST expects to standardise multiple winners so that if one of the standardised algorithms turns out to be insecure there are different options to fall back to.

1.4 Research problems and overview of the thesis

This thesis contains my contributions towards three broad research objectives listed below. These contributions are presented in Part II of this doctoral thesis as a set of research papers. The remainder of Part I of the thesis is split up into three chapters, one for each research objective. Each chapter gives an overview of the relevant state-of-the-art, and explains the contributions I made towards the research objective.

Objective 1: Expanding the set of hard problems we can build post-quantum cryptography on. As mentioned in the introduction, RSA signatures rely on the hardness of factoring integers, and elliptic curve signatures rely on the hardness of the discrete logarithm problem. It is dangerous to have all of cryptography rely on a few hard problems, because if those problems become efficiently solvable then all cryptography becomes insecure, which is precisely what will happen when sufficiently powerful quantum computers are built. The same risk applies to post-quantum cryptography: if all the post-quantum algorithms rely on a small set of hard problems, then we risk losing all our cryptography again when those problems become efficiently solvable (e.g., if another fundamentally different computing technology arises, but more likely because someone discovers an efficient algorithm). An important research objective is therefore to diversify the set of hard problems that we can build post-quantum cryptography from, to avoid putting all our eggs in one basket. Moreover, by researching new hard problems we might find that we can construct more efficient algorithms, or construct advanced cryptographic primitives that we do not know how to build from other hard problems. This research question is discussed in Chapter 2.

Objective 2: Evaluating the (in)security of post-quantum signatures. Currently, there do not exist public-key algorithms that we can unconditionally prove to be secure against attackers with limited computational resources. At best, we can prove that breaking a cryptographic algorithm is at least as hard as solving some other problem that seems difficult to solve, but some cryptographic algorithms (including the RSA algorithm[5]) we can not even prove such a guarantee. Therefore, it is important to study the hardness of the underlying problem, or to study the security of the cryptographic algorithm directly by looking for efficient algorithms that can break the scheme. If after substantial effort no serious vulnerabilities are found, we can have some confidence that the algorithm is secure and can be used in applications. The research objective of finding vulnerabilities in post-quantum signature algorithms is discussed in Chapter 3.

Objective 3: Designing secure and more efficient post-quantum signatures. Currently, most post-quantum signature schemes have significantly larger key and signature sizes or have much slower signing and verification operations than the signatures that are in use today. For example, the current ECDSA keys and signatures are only 32 Bytes and 64 Bytes large, while Falcon [17], the most compact lattice-based algorithm, has keys of 897 bytes and signatures of 666 bytes. SQISign [23], the latest isogeny-based signature algorithm has key and signature sizes that are only approximately twice as large as those of ECDSA, but its signing operation takes 2.5 seconds, which is more than 3 orders of magnitude slower than ECDSA and which is too slow for most applications. These practical limitations can make it too costly for users to make the necessary switch to post-quantum cryptography in resource-constrained applications, which hurts the security of our IT infrastructure. To mitigate this problem, the final research objective of this thesis is to design cryptographic algorithms that can be securely implemented as efficiently as possible. This research question is discussed in Chapter 4.

[5]We know that if you can factor integers you can break the security of RSA, but it is not (unconditionally) proven that breaking RSA is at least as hard as factoring integers. In theory, someone could come up with an efficient method to break the security of RSA which does not use an algorithm to factor integers.

Chapter 2

Post-Quantum Hardness Assumptions

Choose your problems rather than letting them choose you.

— Mark Manson, *The Subtle Art of Not Giving a F*ck*

Public key algorithms rely on the assumed hardness of some computational problems for their security. For example, the RSA system can only be secure if factoring integers is hard. This chapter provides an overview of the hardness assumptions used for constructing post-quantum digital signature schemes. In Sect. 2.1 we briefly discuss why hardness assumptions are necessary. In Sect. 2.2, we give very brief introductions to the five most used families of hardness assumptions in post-quantum cryptography. Finally, in Sect. 2.2.5 we cover my contributions towards expanding the set of hardness assumptions that post-quantum cryptography can be built upon.

2.1 Why hardness assumptions?

Ideally, we would want to use an efficient digital signature algorithm for which we can mathematically prove that without the secret key, an attacker can not produce a valid signature for any message. Unfortunately, this is not possible, because the attacker can just try all the possible signatures until he finds a valid one. The next best thing we can hope for is to prove that forging a signature is computationally expensive. For example, we could hope to prove that any algorithm to forge signatures takes on average at least 2^{128} operations.

Unfortunately, proving that certain computational tasks cannot be solved efficiently is a notoriously difficult problem in computer science. Even for seemingly simple tasks such as multiplying integers computer scientists struggle to prove non-trivial lower bounds. Finding a family of signature schemes for which the verification algorithm is efficient (i.e., can be done with a polynomial-time algorithm), but for which forging a signature is not efficient (i.e., cannot be done with a polynomial-time algorithm) would solve the P $\neq$ NP problem, one of the 7 Millennium prize problems with a bounty of 1 million USD [18]. It should therefore not be surprising that no cryptographer has been able to construct a digital signature algorithm with an unconditional security proof, and it seems unlikely that this will happen in the near future.

Reductions. While it is difficult to rigorously prove that a computational problem A is hard, it can be much easier to prove that problem A is at least as hard as some other problem B. If you can solve instances of problem B by efficiently using an algorithm that solves problem A as a subroutine, then that proves that problem A is at least as hard as problem B. (If A can be solved efficiently, then B can also be solved efficiently by plugging the efficient algorithm for A into the algorithm for B). In computer science, this is called a *reduction*, and we say that problem B can be reduced to problem A.

Example 2.1 (Multiplication is at least as hard as addition). *Most people would agree that multiplying two numbers seems strictly harder than adding 2 numbers, but we currently do not know how to (unconditionally) prove that this is the case [42]. However, it is simple to prove that multiplication is not simpler than addition with a reduction. To do this, we efficiently transform the addition problem into a multiplication problem as follows: suppose we are given two bit strings of length d that represent two numbers $x, y \in \mathbb{N}$ in binary notation. Our task is to compute a binary representation of length $d+1$ of their sum $x+y$. We can efficiently compute a representation of $A = 1 + 2^{d+1}a$ and $B = 1 + 2^{d+1}b$ by putting a '1' and d '0's in front of the representations of a and b.*

If we multiply A and B we get

$$AB = (1 + 2^{d+1}a)(1 + 2^{d+1}b) = 1 + 2^{d+1}(a+b) + 2^{2d+2}ab$$

whose binary representation is

$$< AB >_2 = 1\underbrace{00\ldots00}_{d\ zeros}\underbrace{x_1x_2\ldots x_{d+1}}_{digits\ of\ a+b}\underbrace{y_1y_2\ldots y_l}_{digits\ of\ ab}\ .$$

We have shown that one can add two numbers by multiplying two (slightly larger) numbers and extracting the solution from the product. This means that multiplying is not easier than adding.

Provable security. Since cryptographers have very slim hopes of directly proving that their signature scheme is secure, they use the reduction technique instead. Signature schemes are constructed in such a way that one can reduce some computational problem (ideally a well-studied problem such as the discrete logarithm problem) to the problem of forging signatures. If such a security reduction exists it means that breaking the security of the scheme is at least as hard as solving the computational problem and we say that the scheme is "provably secure", even though the security only follows *from the assumption that the computational problem is hard.* A security proof is not a guarantee that a signature algorithm cannot be broken by attackers, but is at best a tool to make scrutinizing the algorithm easier. It would be too costly to spend tens of thousands of man-hours on analyzing the security of an algorithm before using it in practice[1], but if our algorithms have security reductions from a small set of well-established hardness assumptions it makes the problem of scrutinizing our algorithms more manageable.

2.2 Post-Quantum hardness assumptions

This section gives a short introduction to the five most frequently used families of hardness assumptions in post-quantum cryptography. But first, since we will be looking at some mathematical problems, we will introduce some common mathematical notation.

Notation. For a prime power q, we denote by $\mathbb{F}_q$ the finite field of order q. Matrices will be denoted by upper case letters, e.g., M, and vectors will be denoted by bold lower case letters, e.g., $\mathbf{x}$. We denote by $GL(k, \mathbb{F}_q)$ the group of invertible k-by-k matrices over $\mathbb{F}_q$. We denote by $S(n)$ the group of permutations of a set of size n. Additional notation will be introduced as needed.

[1]IBM claimed to have spent 17 man-years on the cryptanalysis of the Data Encryption Standard (DES), but nevertheless more weaknesses were discovered years after the standard was published (although the weaknesses did not result in practical attacks).

2.2.1 Multivariate quadratic assumption

Multivariate Quadratic (MQ) cryptography is based on the assumed hardness of finding a solution to a system of multivariate quadratic equations over a finite field. This problem is called the MQ problem and is defined more formally as follows.

Definition 2.2 (multivariate quadratic map)**.** A multivariate quadratic map $\mathcal{P}$ with m components in n variables over a field K is a function

$$\mathcal{P} : K^n \to K^m : \mathbf{x} \mapsto \mathcal{P}(\mathbf{x}) = (p_1(\mathbf{x}), \dots, p_m(\mathbf{x}))$$

where $p_1, \dots, p_m$ is a list of m quadratic polynomials in n variables with coefficients in K.

Definition 2.3 (MQ problem)**.** The MQ problem asks, given a multivariate quadratic map $\mathcal{P} : \mathbb{F}_q^n \to \mathbb{F}_q^m$ over a finite field $\mathbb{F}_q$, and a target $\mathbf{t} \in \mathbb{F}_q^m$, to find a solution $\mathbf{s}$ such that $\mathcal{P}(\mathbf{s}) = \mathbf{t}$.

The MQ problem is known to be NP-hard[2] over any finite field $\mathbb{F}_q$ [38], which is a very strong argument for its worst-case hardness. If $n > m(m+1)$ or $m > n(n-1)/2$ the problem can be solved in polynomial time [69], but when $n \sim m$ the problem is believed to be exponentially hard on average, which is important if we want to use random (or random-looking) instances of the MQ problem as a basis for our cryptographic algorithms.

The most efficient algorithms for solving cryptographically relevant instances of the MQ problem are (variants of) F_4, F_5, and XL [34, 34, 19], which rely on algebraic techniques related to Gröbner bases. For parameter sets over small finite fields (e.g. $q = 2, 3$ or 4) the algorithm of Joux and Vitse [44], which is a crossbred between exhaustive search and Gröbner basis methods, can be the more efficient than purely algebraic algorithms.

Table 2.1 shows the estimated gate count of the best classical algorithms for finding a solution to random MQ systems. We see that the complexity increases exponentially with the increasing size of the system (number of variables and number of equations). For fields of size $q = 16$ we see that it suffices to take 60 equations in 60 variables to get an estimated complexity of 2^{155} gates, which seems currently far too computationally expensive for an attacker. This is why the Rainbow signature scheme uses a system with 64 equations over $\mathbb{F}_{16}$ for NIST security level I. In the right half of the table, we see that if the number of equations increases for a fixed number of variables, then the MQ problem can

[2]Informally, a problem X is NP-hard if any problem whose solution can be efficiently verified can be reduced to X.

Table 2.1: An overview of the estimated gate counts of the best classical algorithms for finding a solution to a random system of m multivariate quadratic equations in n variables over a finite field of size $q = 16$.

n	m	log_2(#gates)	n	m	log_2(#gates)
20	20	65	60	60	155
30	30	88	60	70	137
40	40	110	60	80	119
50	50	133	60	90	110
60	60	155	60	100	103
70	70	177	60	110	94
80	80	198	60	120	92

be solved more efficiently. This means that if an attacker can somehow obtain enough additional equations, then an MQ problem might become easy enough to solve.

There are two kinds of signature schemes based on the MQ problem, traditional MQ schemes and the more modern provably secure schemes.

Traditional MQ signatures

The first MQ signature algorithm was the C^* algorithm proposed in 1988 by Matsumoto and Imai [49]. Although the scheme was shown to be insecure by Patarin [56], the design of C^* has inspired many other MQ signature schemes. The central idea goes as follows:

Key generation. The key generation algorithm chooses a multivariate quadratic map $\mathcal{F} : \mathbb{F}_q^n \to \mathbb{F}_q^m$ with some special structure that makes it possible to efficiently solve the MQ problem for $\mathcal{F}$ (the specific structure varies from scheme to scheme). Then, $\mathcal{F}$ is randomized by composing it with 2 random invertible linear maps $\mathcal{S} \in GL(m, \mathbb{F}_q)$ and $\mathcal{T} \in GL(n, \mathbb{F}_q)$. The randomized map $\mathcal{P} = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T}$ is used as the public key, while the factorization $\mathcal{S}, \mathcal{F}, \mathcal{T}$ is the secret key.

Signing. To sign a message m, the user first hashes the message to a digest $\mathbf{t} = \mathcal{H}(m)$, with a collision resistant hash function $\mathcal{H}$ that outputs elements in the co-domain of $\mathcal{P}$ (see Sect. 2.2.3). Then, the signer uses his knowledge of the factorization $\mathcal{T}, \mathcal{F}, \mathcal{S}$ to compute a signature $\mathbf{s}$ such that $\mathcal{P}(\mathbf{s}) = \mathbf{t}$. This can be done efficiently in 3 steps: First compute $\mathbf{t}' = \mathcal{S}^{-1}(\mathbf{t})$, then use the structure of $\mathcal{F}$ to compute $\mathbf{s}'$ such that $\mathcal{F}(\mathbf{s}') = \mathbf{t}'$ and finally compute the signature $\mathbf{s} = \mathcal{T}^{-1}(\mathbf{s}')$.

Verification. To verify if a signature is valid, the verifier simply checks whether $\mathcal{P}(\mathbf{s}) = \mathcal{H}(m)$.

The hope is that composing the special map $\mathcal{F}$ with random linear maps makes it look like a uniformly random MQ map, so that finding preimages is hard without knowledge of the secret structure hidden in $\mathcal{P}$. The majority of the MQ signature schemes follow this approach, including the two MQ-based signature schemes still in the NIST PQC project Rainbow [31, 29] and G*e*MMS [14][3]. The main advantage of multivariate signature schemes is that the signature size is small because a signature consists of a short vector $\mathbf{s} \in \mathbb{F}_q^n$. The drawbacks are that the public keys are really large, because a public key is a list of m polynomials (e.g. $m = 60$), and each polynomial contains $O(n^2)$ coefficients that need to be stored. For example, the parameter set of the Rainbow signature scheme claiming NIST security level I has a signature size of only 66 bytes, but a public key size of 158 KB. Another drawback is that traditional MQ signatures lack meaningful security reductions, which means that it is harder to gain confidence in their security. Over the years, many MQ signature schemes have been proposed, broken, patched, and broken again in a cycle towards more secure (but also more complicated) signature schemes. For example, during the third round of the NIST PQC project, the attacks on the two remaining candidates Rainbow and G*e*MMS have improved by a factor 2^{20} and 2^{25} respectively [8, 28].

Provably secure schemes

MQDSS [62], SOFIA [16], and MUDFISH [9] are three MQ signature schemes with a fundamentally different approach. They enjoy security reductions from the MQ problem for randomly chosen maps $\mathcal{P}$, which results in more confidence in their security. The performance characteristics of these schemes are also very different from the traditional MQ schemes. Traditional MQ schemes are fast, have small signatures and large keys. In contrast, the provably secure MQ schemes are slower, have larger signature sizes but much smaller keys. For example, the MUDFISH scheme has a signature size of 14.4 KB, but a key size of only 38 bytes at NIST security level I. Signing takes 5 ms on a modern processor, which is roughly a factor 200 slower than Rainbow.

[3]Rainbow is one of the three finalist signature schemes, and G*e*MMS is one of the four alternate signature schemes.

2.2.2 Isogeny-based assumptions

Isogeny-based cryptography relies on the hardness of computational problems related to *isogenies between elliptic curves.* We encountered elliptic curves before in the context of traditional elliptic curve cryptography, where a single elliptic curve E is chosen, and the cryptography was based on arithmetic of the points on the curve E. In contrast, isogeny-based cryptography defines a very large class of elliptic curves and cares about how those elliptic curves relate to each other with so-called *isogenies.*

Isogenies between elliptic curves

Definition 2.4. A map ϕ from an elliptic curve E to a (possibly different) elliptic curve E' is an isogeny, if it is given by non-constant rational functions, i.e., a map of the form

$$\phi : E \to E' : (x, y) \mapsto (f(x, y), g(x, y)),$$

where f and g are non-constant rational functions, and if ϕ maps the point at infinity of E to the point at infinity of E'.

If a map satisfies this definition, it is automatically a group homomorphism from the group of points on E to the group of points on E', meaning that $\phi(P + Q) = \phi(P) + \phi(Q)$ for all $P, Q \in E$. We say E and E' are *isogenous* if there exists an isogeny from E to E'. Being isogenous is an equivalence relation because the composition of two isogenies is itself an isogeny and if there exists an isogeny $\phi : E \to E'$ there also is a so-called dual isogeny $\hat{\phi} : E' \to E$ that goes in the opposite direction.

Every isogeny ϕ has a *degree* $\deg(\phi)$. In isogeny-based cryptography, we usually only encounter separable isogenies, for which the degree corresponds to the number of points in the kernel of ϕ (also counting the points defined over the algebraic closure of the field of definition of E). The degree of the composition of two isogenies is the product of their degrees. We refer to Silverman [66] for a good reference on elliptic curves and isogenies between them.

The computational problems that isogeny-based cryptography relies on are variants of the following basic problem: given two isogenous curves E and E' defined over a large prime field $\mathbb{F}_p$, find an isogeny from E to E'.

Isogeny based signature schemes

Isogeny-based cryptography can be divided into two families. The SIDH-family (named after the first scheme SIDH [43]) and the CRS family, named after Couveignes, Rostovstev, and Stolbunov who started this family [20, 61].

SIDH. Schemes in the SIDH family fix a large prime p and make use of the set of *supersingular* elliptic curves over $\overline{\mathbb{F}_p}$ (up to isomorphism). A helpful fact is that all supersingular curves are isomorphic to a curve defined over $\mathbb{F}_{p^2}$, so one never needs to do arithmetic over larger field extensions. There are roughly $p/12$ supersingular curves, and it turns out that all these curves are pairwise isogenous [51]. For a prime l, we say that the supersingular l-isogeny graph is the graph whose vertices are the supersingular elliptic curves, and whose edges correspond to isogenies of degree l. It turns out that the supersingular l-isogeny graph is $l+1$ regular (i.e., each curve has $l+1$ outgoing l-isogenies), and that the graph is an expander graph, which roughly means that a random walk on the isogeny graph will quickly converge to a uniform distribution on the set of supersingular curves. Digital signature algorithms using this setting were proposed by Yoo *et al.* [70] (based on [43]) but were not very efficient. Recently, a new approach that uses quaternions appeared that has remarkably small key and signature sizes [23].

CRS. In the CRS-family, algorithms fix a large prime p, and a set of elliptic curves $\mathcal{E}\ell\ell_p(\mathcal{O},\pi)$ that consists of all the curves defined over $\mathbb{F}_p$ that share a common $\mathbb{F}_p$-endomorphism ring[4] $\mathcal{O}$.

The set $\mathcal{E}\ell\ell_p(\mathcal{O},\pi)$ is chosen because there is a certain commutative group $\mathcal{C}\ell(\mathcal{O})$, called the *ideal class group of the endomorphism ring* $\mathcal{O}$, whose elements are represented by invertible fractional ideals in $\mathcal{O}$, that acts on $\mathcal{E}\ell\ell_p(\mathcal{O},\pi)$. The group action is defined as

$$\star : \mathcal{C}\ell(\mathcal{O}) \times \mathcal{E}\ell\ell_p(\mathcal{O},\pi) \to \mathcal{E}\ell\ell_p(\mathcal{O},\pi) : [\mathfrak{a}] \star E := E/E[\mathfrak{a}]$$

where $E[\mathfrak{a}] = \cap_{\phi\in\mathfrak{a}} \ker(\phi)$ is a finite subgroup of E, and $E/E[\mathfrak{a}]$ is the unique (up to $\mathbb{F}_p$ isomorphism) elliptic curve for which there is an isogeny $\phi : E \to E/E[\mathfrak{a}]$ whose kernel is exactly $E[\mathfrak{a}]$. This group action is free and transitive, which means that for each pair of curves $(E, E') \in \mathcal{E}\ell\ell_p(\mathcal{O},\pi)^2$ there is a unique element $[\mathfrak{a}]$ in the class group for which $[\mathfrak{a}] \star E = E'$.

[4]The $\mathbb{F}_p$-endomorphism ring of an elliptic curve consists of the isogenies from E to itself and the zero morphism. Together these form a ring with pointwise addition and composition of morphisms. We say two elliptic curves E and E' have the same $\mathbb{F}_p$-endomorphism ring if their endomorphism rings are isomorphic, with the isomorphism sending the Frobenius on E to the Frobenius on E'.

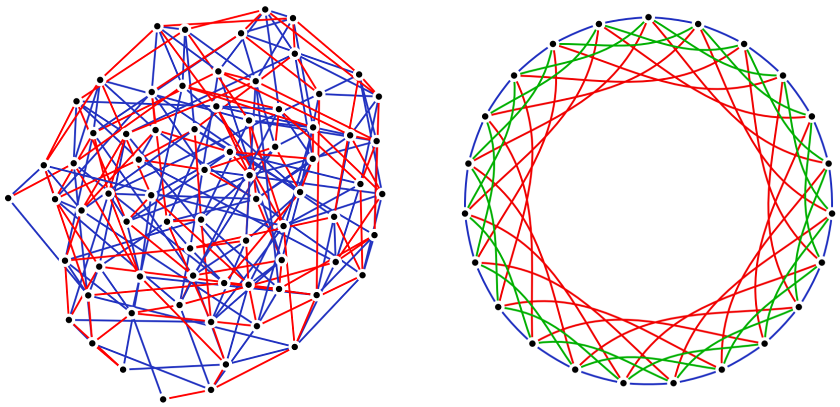


Figure 2.1: Isogeny graphs in the SIDH setting (left) and CRS/CSIDH setting (right). **Left**: Nodes represent supersingular curves mod $p = 863$, 2-isogenies are denoted by red edges, 3-isogenies are denoted by blue edges. (adapted from an image of Enric Florit and Gerard Finol [36]).
Right: Nodes represent supersingular curves defined over $\mathbb{F}_{419}$ with common endomorphism ring $\mathbb{Z}[\sqrt{-419}]$. Red, blue, and green edges correspond to $2, 3$, and 7-isogenies respectively (Image due to Lorenz Panny [15]).

Initially, CRS-like algorithms used ordinary elliptic curves, but Castryck *et al.* discovered that the group action can be evaluated much more efficiently using supersingular elliptic curves instead [15]. The group action is a versatile tool for building cryptographic protocols, including signature schemes [22, 11], but the group action also comes at a cost: there is a generic quantum subexponential-time algorithm that given E and E' can find the ideal class $[\mathfrak{a}]$ such that $[\mathfrak{a}] \star E = E'$. This means that in order to be secure against powerful quantum computers, the prime p used in the CRS setting needs to be asymptotically larger than in the SIDH setting, which can make the scheme less efficient.

2.2.3 Hash-based assumptions

Cryptographic hash functions are functions that can take bitstrings of arbitrary length as input, and output a fixed-length digest. They have many direct applications, such as password hashing and verifying the integrity of messages and files, but they are also frequently used for building more advanced cryptographic algorithms and protocols. In particular, it is possible to build hash-based digital signature schemes, that are secure as long as the underlying hash function has some security property similar to second preimage resistance.

Definition 2.5 (Second preimage resistance (informal))**.** A hash function $\mathcal{H} : \{0,1\}^\star \to \{0,1\}^\lambda$ is second preimage resistant if given an input $\mathbf{x}_1 \in \{0,1\}^\star$ it is hard to find a different input $\mathbf{x}_2 \in \{0,1\}^\star$ such that $\mathcal{H}(\mathbf{x}_1) = \mathcal{H}(\mathbf{x}_2)$.

Cryptographic hash functions such as SHA-3 [33] are well studied and widely believed to be preimage resistant[5], which is why we can have high confidence in the security of hash-based signatures using SHA-3.

On a very high level, a public key for a hash-based signature consists of a hash digest of many secret strings. To sign a message, the signer releases a subset of the secret strings. This is why for some hash-based signatures a user is only allowed to sign a certain number of messages with each secret key because otherwise too much secret information is revealed and the scheme becomes insecure. These schemes are called stateful signature schemes because the signer needs to keep track of a state (i.e., how many signatures have been signed already). Stateless hash-based signatures also exists, these have so much secret information that in reasonable applications the signer will not produce enough signatures to make the scheme insecure. For example, the stateless SPHINCS+ algorithm (one of the algorithms still in the NIST standardization project) is designed to be still secure after signing 2^{64} messages (e.g. 6 trillion signatures per second during the span of 100 years).

2.2.4 Code-based assumptions

Already in 1978, during the early years of public-key cryptography, McEliece proposed the first public-key encryption scheme based on error-correcting codes. Error-correcting codes are a widely used technique that allows one to reliably send messages over an unreliable channel. To do this, the sender transforms the message into a codeword before sending it, and the receiver decodes the codeword to recover the message. Error-correcting codes are useful because even if a limited number of errors is introduced in the codeword, the receiver will still be able to recover the original message.

Most error-correcting codes, including the codes used in code-based cryptography, are linear, meaning that the messages and the codewords are interpreted as vectors over a finite field $\mathbb{F}_q$, and that the encoding function is a linear function. Specifically, if the messages are vectors of length k, and the codeword are vectors of length n, the encoding function is represented by a so-called generator matrix $G \in \mathbb{F}_q^{k\times n}$ with k rows and n columns, and the encoding of a

[5]Preimage resistance is a rather weak requirement of a hash function, usually cryptographers have stronger requirements such as collision resistance, which says it is hard to find two messages $\mathbf{x}$ and $\mathbf{x}'$ such that $\mathcal{H}(\mathbf{x}) = \mathcal{H}(\mathbf{x}')$.

message m is simply $\text{Enc}(m) = mG$. If a random matrix G is used, it seems a computationally hard problem to decode a noisy codeword, and code-based cryptography assumes that this problem is indeed hard.

Definition 2.6 (Generic decoding problem)**.** The generic decoding problem asks, given a generator matrix $G \in \mathbb{F}_q^{k \times n}$ and a noisy codeword $c = mG + e$, where e has small Hamming weight[6], to find the message m.

Even though the generic decoding problem seems computationally hard, coding theorists have worked hard to find specific codes for which very efficient decoding algorithms exist. Goppa codes are one such family of codes, which are also the codes that are used in the McEliece cryptosystem, which goes as follows:

Key Generation. To generate a key pair the user constructs the generator matrix G for a random Goppa code, for which we know there exist efficient decoding algorithms. Then it computes a randomized version of the generator matrix $G' = SGP$ by multiplying it with a random invertible linear map $S \in GL(k, q)$, and a permutation matrix $P \in S_n$. The public key is now G', and the secret key is S, G, and P

Encryption. The encryption of a message m is simply $c = mG' + e$, where $e \in \mathbb{F}_q^n$ is a randomly chosen 'noise' vector, which only has a few nonzero entries. That is, a ciphertext is just the encoding of the message with the generator matrix G' with some noise added to it.

Decryption. If the receiver knows the secret key S, G, P, he can decode the noisy codeword $c \in \mathbb{F}_q^n$ in three steps. First he computes cP^{-1}, which is a noisy encoding of mS under G, because

$$cP^{-1} = (mG' + e)P^{-1} = mSG + eP^{-1} .$$

Second, he uses the efficient decoding algorithm for G to find mS and finally, he recovers the original message by multiplying with S^{-1}.

The hope is that the randomized code G' is indistinguishable from a code chosen uniformly at random because that would mean (assuming decoding in random codes is hard) that an eavesdropper who learns $c = mG' + e$, cannot decode the codeword to recover the message m.

Trapdoor signatures

While code-based encryption is reasonably straightforward (McElieces paper was only 2 pages long), it proved to be more difficult to construct digital

[6]The Hamming weight of a vector is the number of non-zero entries.

signature algorithms from code-based assumptions. A natural approach is to try to sign a message $m \in \mathbb{F}_q^n$ by “decoding” it. A signature for the message would then be a pair $(x, e) \in \mathbb{F}_q^k \times \mathbb{F}_q^n$ with e a vector of small Hamming weight such that $xG' + e = m$. Unfortunately, this is not so easy, because the probability that a message is close enough to a codeword xG in order for the Goppa decoding algorithm to work is very small. Moreover, even if you got this to work, special care needs to be taken to prevent the signature (x, e) from leaking information about the secret key. These obstacles were finally overcome in 2019 [25], resulting in the WAVE signature scheme.

Zero-knowledge-based signatures

There are alternative approaches to build code-based signature schemes, based on non-interactive zero-knowledge proofs of knowledge. A zero-knowledge proof of knowledge allows a prover to convince a verifier that he knows a solution to some problem, without revealing anything about the solution itself. This can be used to create digital signature algorithms as follows:

Key Generation. The user generates a computational problem related to error-correcting codes. The public key is the description of the problem, and the secret key is a solution. For example, the problem could be an instance of the decoding problem. The public key is then a generator matrix G and a noisy codeword $c = mG + e$, where e has small Hamming weight, and the secret key would be m.

Signing. To sign the message, the signer creates a non-interactive zero-knowledge proof of knowledge of a solution to the problem, while incorporating the message to be signed in the proof in such a way that altering the message would invalidate the proof.

Verification. To verify the signature, the user simply checks if the proof is valid.

The zero-knowledge property guarantees that signing messages does not leak information about the secret key, and if the proof system is sound, then this implies that it is impossible to forge signatures without knowing the secret key. Zero-knowledge-based signatures typically have small public keys, but large signature sizes and slow signing times. Examples are the Stern scheme [68], which is based on the decoding problem, and the LESS scheme [12], which is based on the code equivalence problem instead.

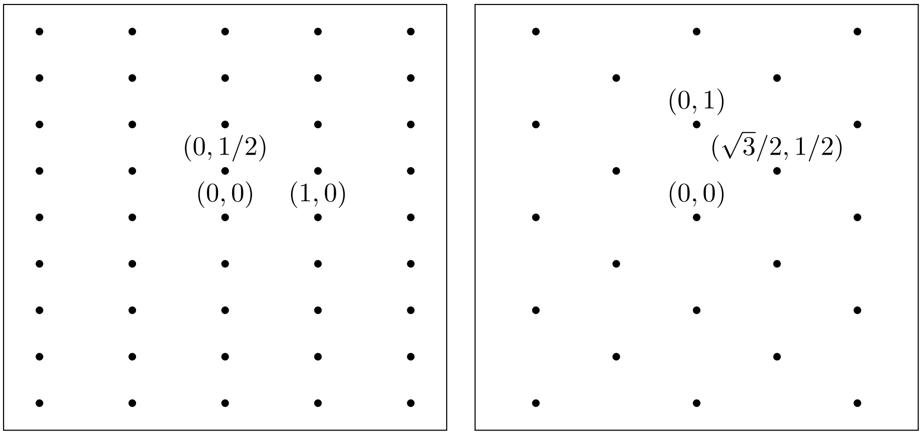


Figure 2.2: This picture shows parts of two 2-dimensional lattices. The left lattice consists of all integer combinations of $(1, 0)$ and $(0, 1/2)$, the right lattice is consists of the integer combinations of $(0, 1)$ and $(\sqrt{3}/2, 1/2)$.

2.2.5 Lattice-based assumptions

Last but not least, we consider lattice-based problems. A lattice is a discrete subgroup of $\mathbb{R}^n$, which means a non-empty set of vectors $L \subset \mathbb{R}^n$ such that:

- if you add or subtract[7] any two vectors in L you get a vector in L (subgroup), and
- there is a minimum distance $d > 0$, [8] such that any two vectors in the lattice are at least at distance d from each other (discrete).

Figure 2.2 shows two lattices of dimension 2. Given a lattice L, we can ask which vector is the closest to 0 (except zero itself). This is called the Shortest Vector Problem (SVP). This problem might seem very easy to solve. Looking at Fig 2.2, it is easy to spot the two shortest vectors in the lattice on the left $((0, \pm 1/2)$), and the 6 shortest vectors in the lattice on the right $((0, \pm 1)$ and $(\pm 1/2, \pm\sqrt{3}/2))$. However, perhaps surprisingly, the problem seems to become very difficult when the dimension increases. The problem even remains hard if we relax the problem to finding a vector that is "not too much" longer than the shortest vector (a problem known as approximate SVP).

[7]Addition and subtraction are performed coordinate-wise, e.g., $(0, 0, 1.5) - (1, 0, 0.5) = (-1, 0, 1)$.

[8]We use the usual Euclidean distance $d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$.

Cryptographers typically do not build cryptography directly on the SVP problem. Instead, they use the Learning with Errors (LWE) and/or Short Integer Solutions (SIS) problem, which have reductions from approximate SVP, which means LWE and SIS they are at least as hard to solve as approximate SVP [2]. The LWE and SIS problems are defined as follows:

Definition 2.7 (LWE)**.** The learning with errors problem asks to learn a vector $\mathbf{s} \in \mathbb{Z}_q^n$, given access to arbitrarily many noisy linear combinations of the entries of $\mathbf{s}$. More precisely, to find $\mathbf{s}$ you are given arbitrarily many samples of the form $(\mathbf{a}_i, b_i = \mathbf{a}_i \cdot \mathbf{s} + e_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$, where the $\mathbf{a}_i \in \mathbb{Z}_q^n$ are chosen uniformly at random and where the errors e_i are drawn from some known error distribution χ.

Definition 2.8 (SIS)**.** The short integer solution problem asks to find a non-zero integer solution $\mathbf{s} \in \mathbb{Z}^n$ to a system of linear equations $\mathbf{s}A = \mathbf{t} \mod q$, where A is a matrix in $\mathbb{Z}_q^{n \times m}$ with $m > n$, such that the solution $\mathbf{s}$ has short length, i.e., $||\mathbf{s}||_2 \leq \beta$, for some bound $\beta \in \mathbb{R}$.

Lattice-based digital signatures

Similar to MQ and code-based cryptography, there are two methods of constructing lattice-based signature schemes: one can use lattice trapdoors or zero-knowledge proofs.

In a trapdoor-based approach, the signer somehow generates a matrix $A \in \mathbb{Z}_q^{n \times m}$ for which only the signer can solve the SIS problem, but such that for outsiders the SIS problem remains hard. To sign a message $\mathbf{t} \in \mathbb{Z}_q^n$, the signer uses the trapdoor to find a solution $\mathbf{s}$ such that $\mathbf{s}A = \mathbf{t}$. Care needs to be taken to make sure that using the trapdoor does not leak the secret key. Falcon [58], one of the three remaining finalists in the NIST PQC project, is a trapdoor-based lattice signature scheme with relatively compact signatures (666 bytes at SL I) and key size (897 bytes) and good signing and verification performance (e.g. more than 10 000 signatures per second on a modern CPU).

In the zero-knowledge-based approach, the signer creates an instance of a lattice problem for which only the signer knows the solution, (for example, an LWE instance, consisting of a number of samples $(\mathbf{a}_i, b_i = \mathbf{a}_i \cdot \mathbf{s} + e_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q)$, and to sign a message he creates a zero-knowledge proof to prove that he knows the solution. Dilithium [48], another finalist in the NIST PQC project, is a zero-knowledge-based lattice signature scheme. Dilithium also has good performance, but slightly larger signature and key sizes compared to Falcon, for example 1.3 KB keys and 2.4 KB signatures at the lowest security level.

Contribution 1: Diversifying hardness assumptions

The previous section shows that there are post-quantum digital signature schemes based on a variety of assumptions. However, despite substantial progress in isogeny-based and code-based signatures, only the lattice-based, hash-based, and multivariate-based signatures are currently considered to be somewhat mature and practical, and even these signature schemes are less practical than the existing RSA and ECC signatures that are in use today in terms of key and signature sizes. The security of hash-based signatures seems to be sound, but the security of lattice-based and especially multivariate signatures seems to be on shakier grounds, as indicated by the recent attacks on both of the remaining multivariate signature schemes still in the NIST PQC project. Therefore, it is important to keep looking for new assumptions and new ways of building digital signature algorithms, in the hope of finding more efficient algorithms, and to have viable options to fall back to when lattice-based and/or multivariate signatures are found to be insecure.

This section of the thesis deals with my contributions towards the objective of diversifying the set of hardness assumptions from which we can build post-quantum digital signatures. I designed two digital signature algorithms based on the assumed hardness of the Permuted Kernel Problem, which is an old combinatorial problem that was first proposed to be used in cryptography in '89 but never gained much traction. In collaboration with my co-authors, I also investigated the security of the Legendre PRF, a pseudo-random function family based on the Legendre symbol and higher-power residue symbols. Then, I designed the first digital signature scheme based on the security of the Legendre PRF and higher-power residue character PRFs.

C1.A Permuted Kernel Problem.

The Permuted Kernel Problem is similar to the SIS problem and the decoding problem in the sense that the goal is to find a solution $\mathbf{s}$ to an underdetermined system of linear equations $\mathbf{s}A = \mathbf{t}$ that satisfies some additional constraints. In the SIS problem, the additional constraint is that the solution has small Euclidean norm $||\mathbf{s}||_2 \leq \beta$. To decode a codeword $\mathbf{c} = \mathbf{m}G + \mathbf{e}$, we look for the error vector $\mathbf{e}$ that satisfies $H\mathbf{e} = H\mathbf{c}$, where H is the parity check matrix of the code. In other words, the problem is to find a solution to the linear system $H\mathbf{e} = H\mathbf{c}$, with the additional constraint that $\mathbf{e}$ has low Hamming weight. In the Permuted Kernel Problem, the task is to find a solution $\mathbf{s}$ to $\mathbf{s}A = 0$ that satisfies the additional constraint that $\mathbf{s}$ is a permutation of a known vector $\mathbf{v}$ (i.e., that entries of $\mathbf{s}$ are the same as the entries of $\mathbf{v}$, but in a different order).

For a vector $\mathbf{v}$ of length n and a permutation $\pi \in S_n$, we denote by $\mathbf{v}_\pi$ the vector obtained by applying π to the entries of $\mathbf{v}$, i.e., $\mathbf{v}_\pi = [v_{\pi(i)}]_{i\in[n]}$.

Definition 2.9. Given a matrix $A \in \mathbb{F}_q^{n\times m}$ and a vector $\mathbf{v} \in \mathbb{F}_q^n$, the Permuted Kernel Problem (PKP) asks to find a permutation $\pi \in S_n$, such that $\mathbf{v}_\pi A = 0$.

Shamir proposed to use the assumed hardness of the permuted kernel problem as a basis for an identification scheme in 1989 [64], but neither the scheme nor the hardness assumption gained much traction, presumably because the scheme was not competitive with RSA signatures. No efficient quantum algorithms are known for solving the permuted kernel problem, and since the permuted kernel problem is NP-hard, it is considered very unlikely that an efficient worst-case quantum algorithm exists. This makes the permuted kernel problem suitable for post-quantum cryptography.

PKP-DSS

Chapter 6 presents a research paper I coauthored which revisits Shamir's identification protocol from 1989. We present concrete parameter choices for the permuted kernel problem, and we improve Shamir's protocol to reduce the key size and the communication size. We then turn this identification protocol into a digital signature algorithm with the Fiat-Shamir transform [35]. This resulted in the PKP-DSS signature scheme with a key size of 20.4 KB, and a key size of 59 bytes. We also provided a constant-time software implementation of PKP-DSS with reasonably good performance (± 1 ms for a signing operation on a modern CPU). We submitted PKP-DSS to the post-quantum competition organised by the Chinese Association for Cryptology Research (CACR), where it won a third prize worth 60 000 Chinese yuan.

SUSHSYFISH

I designed a more efficient zero-knowledge protocol for proving knowledge of a solution to a PKP instance in my paper of Chapter 14. Shamir's identification protocol uses 5 rounds of communication and has a soundness error of $\frac{q}{2q-2} \approx 1/2$, which means that the protocol must be repeated roughly 128 times to get 128 bits of security. The FS transform for 5 round protocols does not perfectly preserve soundness, which means that to make a signature scheme with 128 bits of security even more repetitions are needed (e.g. 157 repetitions in the case of PKP-DSS).

My new protocol has a soundness error of $1/q$, which means that the protocol only needs to be repeated $128/\log(1/q)$ times. Moreover, the new protocol uses only 3 rounds of communication, which means no additional iterations are required to keep the non-interactive signature scheme secure. While an execution of the new protocol is more expensive than Shamir's protocol, the reduced number of iterations makes the new protocol more efficient overall. The new signature scheme is called SUSHSYFISH, and has a signature size of 12.1 KB and a key size of 72 bytes (to be compared with 20.4 KB and 59 bytes for PKP-DSS). The new protocol follows the sigma-protocol-with-helper framework that I introduced, and that will be discussed in more detail in the Section 4.3

C1.B Legendre PRF.

Cryptanalysis of the Legendre PRF

The Legendre Pseudo-Random Function (PRF) is a relatively unknown and little-used piece of cryptography based on the Legendre symbol, which was introduced by Legendre in 1798. For each odd prime p, the Legendre symbol mod p is the function

$$\left(\frac{a}{p}\right) = \begin{cases} 1 & \text{if } a \text{ is a quadratic residue mod } p \text{ and } a \neq 0 \mod p\,, \\ -1 & \text{if } a \text{ is not a quadratic residue mod } p\,, \\ 0 & \text{if } a = 0 \mod p\,. \end{cases}$$

A sequence of Legendre symbols $\left(\frac{k}{p}\right), \left(\frac{k+1}{p}\right), \left(\frac{k+2}{p}\right), \dots, \left(\frac{k+l}{p}\right)$ looks very much like a uniformly random sequence of 1s and -1s, which is why in 1988 Damgård proposed to use the Legendre symbol to construct a pseudo-random function family. Concretely, he proposed to use the set of function $\{L_K\}_{K \in \mathbb{F}_p}$, defined as

$$L_K(x) := \left\lfloor \left(\left(\frac{a+K}{p} \right) + 1 \right) /2 \right\rfloor .$$

This family of functions is conjectured to be pseudo-random, meaning that it is hard to distinguish the input-output behavior of L_K for a random choice of $K \in \mathbb{F}_p$ from a uniformly random function from $\mathbb{F}_p$ to $\{0, 1\}$.

In the paper of Chapter 7, I investigated the security of this PRF in collaboration with Beyne, Udovenko, and Vitto. We found new attacks that given access to a function L_K, can recover the key K. If we are allowed to make M queries to the PRF, our attack runs in time $O(M^2 + p\log^2 p/M^2)$, which is much better than the best previously known attack which took time $O(M + p\log p/M)$. Initially, our interest in the Legendre PRF was motivated by the fact that the Ethereum foundation was planning to use the Legendre PRF in their Ethereum 2.0 proof-of-custody mechanism, and that they were awarding bounties to whomever could find more efficient attacks and solve challenges[9]. However, the Legendre PRF (and its generalisation to higher-power residues) also turned out to be useful as a basis for efficient post-quantum signatures.

Signature schemes based on the Legendre PRF

I designed LegRoast and PorcRoast, two MPC-in-the-head signature schemes based on the assumed one-wayness of the Legendre PRF, and the power-residue PRF, which were published in the paper of Chapter 8, which I co-wrote with Delpech de Saint Guilhem. The secret key for LegRoast is a value $K \in \mathbb{F}_p$, and the public key consists of the evaluation of L_K at a small number of publicly known inputs $a_1, \ldots, a_L$. We prove that LegRoast is secure if given the evaluations $L_K(a_1), \ldots, L_K(a_L)$, it is hard to find the key $K \in \mathbb{F}_p$, which is a weaker assumption than the security of the Legendre PRF. LegRoast is the first digital signature scheme based on this assumption, has a key size of 4 KB, and a signature size of 12.2 KB. PorcRoast is a variant of LegRoast that relies on the hardness of the same computational problem if we replace the Legendre symbol with a higher-power residue symbol. That is, instead of $\left(\frac{a}{p}\right)$ (which is equal to $a^{(p-1)/2} \mod p$), we use the k-th power residue symbol

$$\left(\frac{a}{p}\right)_k := a^{(p-1)/k} \mod p\,,$$

for some integer k, and a large prime $p = 1 \mod k$. In comparison to LegRoast, PorcRoast has smaller keys and signatures, for example 0.5 KB keys and 7.4 KB signatures at NIST SL 1.

2.3 Conclusion

In this chapter, we saw that there exists a large number of post-quantum digital signature schemes whose security is based on a variety of hardness assumptions. The most important and widely used hardness assumptions are related to multivariate systems of polynomials, error-correcting codes, isogenies between elliptic curves, hash functions, and lattices. These assumptions give rise to reasonably efficient signature schemes, but there still is a significant gap between the performance of post-quantum schemes and the RSA and ECC

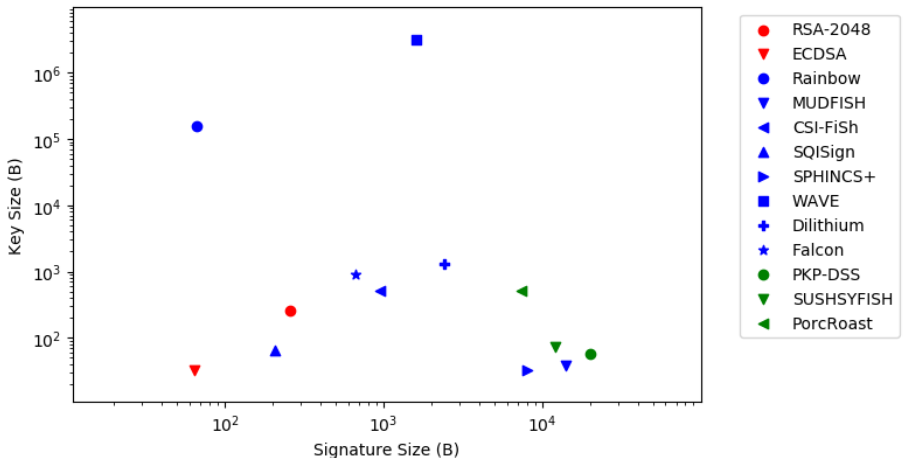
[9]Our work led to 2000 USD and 3 ETH in prize money.

digital signatures that are in use today in terms of key and signature sizes (see Table 2.2). An exception is the SQISign scheme, but currently, this scheme has slow signing times and its security is not yet fully understood. Therefore, it is good to keep investigating new assumptions, in the hope that they can be the basis of more efficient post-quantum signature algorithms.

This chapter briefly introduces two such ‘new’ hardness assumptions: the hardness of the Permuted Kernel Problem and the security of the Legendre PRF. I cryptanalysed the LegendrePRF in Chapter 7, and I designed three signature schemes based on these assumptions: PKP-DSS, SUSHSYFISH, and LegRoast/PorcRoast. The designs are published in the papers in Chapters 6,14, and 8 respectively. Currently, the performance of these new schemes is comparable to existing post-quantum signature schemes. Future research will tell if the signature schemes can be made more efficient, or if the assumptions can be attacked more efficiently, in which case larger parameters are required, leading to less efficient signature schemes

Table 2.2: A list of hardness assumptions we can build digital signature algorithms from. The assumptions are listed in three groups: first, we have two assumptions that are not post-quantum secure, then the five most common post-quantum hardness assumptions, followed by two new post-quantum assumptions. For each hardness assumption, we chose one or two digital signatures, targeting 128 bits of security, to give an idea of how large the public keys and signatures typically are.

Hardness Assumption	Signature scheme	Public Key	Signature
Factoring	RSA-2048	256 B	256 B
DLOG	ECDSA (256 bit)	32 B	64 B
MQ	Rainbow	158 KB	66 B
	MUDFISH	38 B	14 KB
Isogenies	CSI-FiSh	512 B	956 B
	SQISign	64 B	207 B
Hash functions	SPHINCS+	32 B	8 KB
Codes	WAVE	3.2 MB	1.6 KB
Lattices	Dilithium	1.3 KB	2.4 KB
	Falcon	897 B	666 B
Permuted Kernels	PKPDSS	57 B	20 KB
	SUSHSYFISH	72 B	12 KB
Legendre Symbols	PorcRoast	0.5 KB	7.4 KB



Chapter 3

Cryptanalysis

Codemaking is a science, and codebreaking is an art.

– Adi Shamir. RSAC 2015 Cryptographers' panel

The previous chapter mentioned security reductions, which can be a useful tool for studying the security of a cryptographic algorithm. But ultimately, the only way to gain confidence in the security of a cryptographic algorithm is by subjecting the algorithm to the analysis of the best cryptographers around. If after many years no vulnerabilities or attacks are found, then it is plausible that the scheme is secure and suitable to be used in practice. This scrutiny of cryptographic algorithms is called cryptanalysis.

This chapter will introduce my contributions towards the cryptanalysis of post-quantum digital signature schemes, but first, we need to formally define what it means for a digital signature scheme to be (in)secure.

3.1 Security definitions for signature schemes

3.1.1 Notation, challengers, and adversaries

Notation. If X is a finite set, we write $x \leftarrow X$ to denote that x is sampled uniformly at random from X. If $\mathcal{A}$ is a (randomized) algorithm, we write $y \leftarrow \mathcal{A}(x)$ to denote that y is assigned the output of $\mathcal{A}$, after running it on input x. If $\mathcal{A}$ is an interactive algorithm, then $y \leftarrow \mathcal{A}^{\mathcal{B}}(x)$ means that y is the output of $\mathcal{A}$ after running it on input x, and interacting with algorithm $\mathcal{B}$.

Challengers and Adversaries. A security property usually demands that a certain task can not be performed by a certain type of algorithm. These statements are formalized with a *challenger* algorithm $\mathcal{C}$ that interacts with an adversary $\mathcal{A}$ and outputs 1 if $\mathcal{A}$ succeeds in doing the task, and outputs 0 if $\mathcal{A}$ fails. The security property is then usually a statement of the form "For every adversary $\mathcal{A}$ in some class of algorithms, $\Pr[\mathcal{C}^{\mathcal{A}}(\lambda) = 1]$ is *small*", where smallness can be defined in a number of ways. This thesis mostly deals with concrete security, so we will opt for concrete security definitions, where we say a primitive has *n bits of security* (or 2^n-security), if every algorithm that runs in time t, [1] has a success probability $\Pr[\mathcal{C}^{\mathcal{A}}(\lambda) = 1]$ of at most $2^n/t$.

3.1.2 Universal unforgeability under key-only attacks

A signature scheme $S = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ is insecure if there exists an efficient algorithm that, given a message m and a public key pk, outputs a signature such that $\mathsf{Verify}(m, \mathsf{sig}, \mathsf{pk}) = \mathsf{accept}$. Conversely, we might be tempted to say that a signature scheme is secure if no such algorithm exists. To formalize this first attempt, we define a challenger algorithm that generates a public key pk using KeyGen, picks a random message m, gives both to the adversary $\mathcal{A}$ and outputs 1 only if the output of $\mathcal{A}$ is a valid signature for the message m under the public key pk.

```
$\mathcal{C}^{\mathcal{A}}_{S,UUF-KOA}$ :
    $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}()$
    $m \leftarrow \{0,1\}^{128}$
    $\mathsf{sig} \leftarrow \mathcal{A}(\mathsf{pk}, m)$
    **return** $\mathsf{Verify}(\mathsf{pk}, m)$
```

We can then say a digital signature scheme is secure if there are no efficient algorithms that win this game with a large probability. This security notion is known as UUF-KOA security, which stands for universal unforgeability under key-only attacks.[2]

Definition 3.1 (UUF-KOA)**.** A signature scheme $S = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ has n bits of UUF-KOA security, if for every $\mathcal{A}$ that runs in time t and has success probability $\Pr[\mathcal{C}^{\mathcal{A}}_{S,UUF-KOA}() = 1] = \epsilon$, we have $\frac{t}{\epsilon} > 2^n$.

[1] We implicitly assume some model of computation, such as the RAM model.

[2] Universal refers to the fact that the attacker can forge a signature for any message, and key-only refers to the fact that the attacker only learns the public key, but no additional information.

UUF-KOA is a very weak security property, and is generally not sufficient in practice. To illustrate this, consider the following toy signature scheme, which uses a hash function $\mathcal{H} : \{0,1\}^\star \to \{0,1\}^{128}$, which we assume to be preimage resistant:

- $\mathsf{KeyGen}()$: Pick a random bit string sk of length 128 as secret key, and let $\mathsf{pk} = \mathcal{H}(\mathsf{sk})$ be the public key.
- $\mathsf{Sign}(m, \mathsf{sk})$: Return sk as a signature.
- $\mathsf{Verify}(m, \mathsf{sig}, \mathsf{pk})$: Accept the signature if $\mathcal{H}(\mathsf{sig}) = \mathsf{pk}$, regardless of the message m.

In the UUF-KOA game, the adversary is given a public key pk, and to win the game $\mathcal{A}$ needs to output a string sig such that $\mathcal{H}(\mathsf{sig}) = \mathsf{pk}$. If we assume that there are no algorithms for finding preimages in $\mathcal{H}$ that are better than a brute-force search, then this toy signature scheme has 128 bits of UUF-KOA security. But the toy signature scheme is totally insecure in practice! An attacker can arbitrarily change a signed message without invalidating the signature. This attack is not captured by the UUF-KOA security definition, which means we need stronger definitions.

3.1.3 Existential unforgeability under chosen message attacks

We can strengthen the UUF-KOA definition in two ways. Firstly, we will assume that the adversary has access to an oracle that will sign arbitrary messages for him. It is not completely realistic that an attacker has access to such a service, but in practice, the attacker is often able to obtain signatures for messages that he can partly choose. For example, by browsing to a secure website, a person can obtain a signature on a session id that the user can influence. An attack that makes use of such a service is called a chosen message attack. Secondly, instead of letting the challenger choose the message that $\mathcal{A}$ has to forge a signature for (i.e., universal forgery), we will let $\mathcal{A}$ choose a message himself (this is called an existential forgery). The adversary does not win the security game if he outputs a signature for a message that he signed using the signing oracle, otherwise, it would be trivial to win the security game and the security definition would be useless. With these enhancements we can formulate the EUF-CMA challenger $\mathcal{C}_{S,EUF-CMA}$ as follows.

```
$\mathcal{C}^{\mathcal{A}}_{S,EUF-CMA}$ :
    $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}()$
    $\mathcal{M} \leftarrow \{\}$
    $(m, \mathsf{sig}) \leftarrow \mathcal{A}^{\mathcal{O}_{\text{Sign}}}(\mathsf{pk})$
    return $\mathsf{Verify}(\mathsf{pk}, m)$ and $m \notin \mathcal{M}$

$\mathcal{O}_{\text{SIGN}}(m)$ :
    $\mathcal{M} \leftarrow \mathcal{M} \cup \{m\}$
    return $\mathsf{Sign}(\mathsf{pk}, m)$
```

Definition 3.2 (EUF-CMA)**.** A signature scheme $S = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ has n bits of EUF-CMA security, if for every $\mathcal{A}$ that runs in time t and has success probability $\Pr[\mathcal{C}^{\mathcal{A}}_{S,EUF-CMA}() = 1] = \epsilon$, we have $\frac{t}{\epsilon} > 2^n$.

EUF-CMA is the most commonly used security definition for digital signature schemes, although even stronger definitions exist, such as strong unforgeability under chosen message attacks (SUF-CMA).

Strong unforgeability. The security game of SUF-CMA is as follows:

```
$\mathcal{C}^{\mathcal{A}}_{S,SUF-CMA}(\lambda)$ :
    $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\lambda)$
    $\mathcal{Q} \leftarrow \{\}$
    $(m, \mathsf{sig}) \leftarrow \mathcal{A}^{\mathcal{O}_{\text{Sign}}}(\mathsf{pk})$
    return $\mathsf{Verify}(\mathsf{pk}, m)$ and $(m, \mathsf{sig}) \notin \mathcal{Q}$

$\mathcal{O}_{\text{SIGN}}(m)$ :
    $\mathsf{sig} \leftarrow \mathsf{Sign}(\mathsf{pk}, m)$
    $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(m, \mathsf{sig})\}$
    return $\mathsf{sig}$
```

The game is identical to the EUF-CMA game, except that the adversary is allowed to output a message-signature pair (m, sig) for a message m on which he queried the signing oracle, as long as sig is not the signature returned by the signing oracle. For example, an attacker who queries $\mathcal{O}_{\text{Sign}}$ for a signature sig on a message m, and output a modified signature sig' that is still valid for the same message m is a valid SUF-CMA attack, but not a valid EUF-CMA attack.

3.2 NIST Security Levels

The above definitions depend heavily on the model of computation that is used. This is problematic because a large number of cost models are used in practice. Some authors count the number of bit operations or gates, while others count the number of multiplications or the number of calls to a cryptographic primitive such as a block cipher. Some authors neglect the cost of accessing memory, while others assign a cost of $c_1 \log(M)$ or even $c_2\sqrt{M}$ to each access to a memory element of size M. This means that n bits of security could mean widely different things, depending on who you ask.

In an attempt to provide a common yardstick for measuring the security of submissions to the PQC competition, NIST defined five security levels. These

Table 3.1: An overview of the five NIST PQC security levels, with the symmetric primitive they are based on, and estimates of the gate count and quantum gate count of the attacks [55], where $d_{max} < 2^{96}$ denotes the maximal allowed depth of the attacking circuit.

Security Level	Symmetric primitive	classical gates	quantum gates
I	128-bit block cipher	2^{143}	$2^{170}/d_{max}$
II	256-bit hash function	2^{146}	
III	192-bit block cipher	2^{207}	$2^{233}/d_{max}$
IV	384-bit hash function	2^{210}	
V	256-bit block cipher	2^{272}	$2^{298}/d_{max}$

security levels are based on the security of symmetric-key primitives. For example, the first security level is defined as "Any attack that breaks the relevant security definition must require computational resources comparable to or greater than those required for key search on a block cipher with a 128-bit key (e.g., AES128)". Security levels I, III, and V are based on a key search against 128, 192, and 256-bit block ciphers respectively, and Security levels II and IV are based on a collision search on a 256 or 384-bit hash function respectively. Table 3.1 contains NIST's estimates of the gate count of a classical or quantum attack against the five security levels [55]. Since Grover's algorithm does not parallelize well, the gate count of a quantum attack depends on the maximal allowed depth d_{max} of the circuit. If the attacker is limited to shallower circuits, then the attack will require more gates. No estimates are provided for the size of a quantum circuit attacking security levels II and IV because there currently are no known quantum algorithms for finding hash collisions that outperform classical algorithms (in realistic cost models).

3.3 Implementation Security

Until now, our focus was entirely on theoretical security properties of abstract digital signature algorithms, but in practice, these algorithms need to be implemented by some physical device, and then the security of the implementation becomes as important as the security of the algorithm itself. Unfortunately, a correct implementation of a secure algorithm is not automatically secure. For example, it is common that the signing algorithm runs faster or slower if a certain situation occurs during the execution of the signing algorithm. Even though the implementation will output signatures correctly, an attacker can pay attention to this timing variability to learn what happened during the execution of the algorithm, which can leak enough information to make the system insecure. Timing information and other types of side-channels (e.g., power consumption or cache usage) often lead to vulnerabilities that can be exploited in practice (see e.g., [45, 46]), unless the algorithm and/or its implementation is carefully designed to prevent leakages. A thorough discussion about implementation security and side-channel attacks is beyond the scope of this thesis.

Contribution 2: Cryptanalysis of post-quantum signature schemes

This section introduces three of my contributions to the cryptanalysis of post-quantum digital signature algorithms. It was very convenient that the beginning of my PhD research aligned with the start of the NIST PQC standardization project because this meant that there was suddenly a large number of post-quantum digital signature schemes with concrete parameter sets and reference implementations whose security needed to be analysed.

C2.A Cryptanalysis of WalnutDSA

WalnutDSA is one of the digital signature algorithms submitted to the NIST PQC standardization project [5]. WalnutDSA is owned by Veridify, a corporation that develops and licenses public-key cryptosystems for low resource processors. Veridify aims for widespread adoption of WalnutDSA in the Internet-of-Things and automotive industries, which makes WalnutDSA a particularly important target for cryptanalysis.

WalnutDSA is not based on the hardness of any of the hard problems described in Chapter 2, but instead on mathematical problems related to *braid groups*. A braid is a mathematical object that describes a collection of intertwined strings. Braids can be inverted, and braids with the same number of strings can be concatenated, which makes the set of braids with n strings (up to equivalence) into a group B_n.

In the paper of Chapter 9, I showed that WalnutDSA (as it was submitted to NIST) is totally insecure. The paper describes three distinct practical attacks against the signature scheme. The most powerful of the three attacks is a UUF-KOA attack, meaning that the attack can forge signatures for arbitrary messages, given only a public key. For the NIST security level I parameters of WalnutDSA, the attack runs in just 1 second on a modern CPU. (Needless to say that this is much faster than a key search on a 128-bit block cipher, which even the most powerful supercomputers in existence today could not complete during the lifetime of the Sun.) For the NIST level V parameter set the attack runs in one minute. NIST did not select WalnutDSA to proceed to the second round of the NIST PQC project, but Veridify is still pursuing market adoption for updated versions of WalnutDSA through partnerships with manufacturers like Intel and STMicroelectronics and by providing free toolkits for popular low-end platforms.

C2.B Cryptanalysis of Code Equivalence Problems

LESS [12] is a new code-based signature scheme. Unlike most code-based cryptosystems, LESS does not rely on the hardness of the generic decoding problem. Instead, it relies on the assumed hardness of Code Equivalence Problems. These problems ask, given two permutationally (resp. linearly) equivalent $\mathbb{F}_q$-linear codes C_1 and C_2, to find a permutation $\pi \in S_n$ (or monomial map[3] $\mu \in M_n$) such that $\pi(C_1) = C_2$ (or $\mu(C_1) = C_2$ respectively).

A public key for LESS is a pair of equivalent codes C_1, C_2, and the corresponding secret key is the equivalence π (or μ) such that $\pi(C_1) = C_2$ (or $\mu(C_1) = C_2$). This means that if one can solve the code equivalence problems, one can recover the secret key from a public key and use it to sign arbitrary messages (this is a UUF-KOA attack). Conversely, the authors of LESS proved that the code equivalence problems can be reduced to the security of the LESS scheme (i.e., they showed that if one can break LESS, one can also solve the code equivalence problems efficiently). This means that LESS is secure if and only if the code equivalence problem is hard, which is exactly what I investigated in my paper of Chapter 10.

I found new algorithms to solve the code equivalence problems which are more efficient than the existing algorithms in the setting used by the LESS cryptosystem. The algorithms work very well in practice: It can recover the secret key of LESS-I (which uses linear equivalence) and LESS-III (which uses permutation equivalence) in 25 seconds and 2 seconds respectively, which stands in sharp contrast to the claim that LESS-I and LESS-III provide 128 bits of EUF-CMA security. The next version of LESS is in the making, which will use larger codes to make the code equivalence problem more difficult. Even though using larger codes will hurt performance, the next version of LESS can still be very competitive with existing code-based signature schemes.

[3]A monomial map is a linear isometry for the Hamming metric. One can show that these maps are the composition of a permutation and a map that multiplies all the entries of a vector by (possibly distinct) non-zero values.

C2.C Cryptanalysis of UOV and Rainbow

One of the three remaining candidates of the NIST PQC project is Rainbow, a multivariate quadratic signature scheme. Rainbow was proposed in 2005 by Ding and Schmidt [31] as a more efficient variant of the UOV signature scheme that was invented in 1997 by Patarin [57]. UOV and Rainbow are both trapdoor-based schemes, where the public key is a trapdoored multivariate quadratic map $\mathcal{P} : \mathbb{F}_q^n \to \mathbb{F}_q^m$, and a signature (s, salt) is valid for the message m if $\mathcal{P}(s) = \mathcal{H}(m, \mathsf{salt})$. The cryptanalysis of UOV and Rainbow has been stable since 2008, which seemingly makes them good candidates for standardization.

However, I discovered two new attacks against the UOV and Rainbow signature schemes. These attacks are described in detail in the paper of Chapter 11. The first attack, which works against both UOV and Rainbow, constructs a system of overdetermined quadratic equations (i.e., the system has more equations than variables). The new system is guaranteed to have a solution, and finding the solution will reveal enough information about the secret key to make a key recovery possible. Since solving overdetermined systems is generally more efficient than solving underdetermined systems, this attack is more efficient than finding a signature s by solving the system $\mathcal{P}(s) = \mathcal{H}(m, \mathsf{salt})$ directly.

The second attack, which only works against the Rainbow scheme, reduces key recovery to the MinRank problem. The attacker converts the public key $\mathcal{P}$ into a list of n-by-m matrices $L_1, \cdots, L_n$, in such a way that there are guaranteed to be linear combinations $\sum_i y_i L_i$ that have exceptionally low rank. Moreover, if one can find one of those linear combinations this gives enough information to efficiently recover the secret key. The problem of, given a list of matrices, finding a low-rank linear combination is called the MinRank problem. This problem has been studied extensively because its hardness is important for both multivariate-based and code-based cryptosystems. Recently an improved method for solving the MinRank problem was discovered [7]. Using this method to solve the MinRank instance derived from the Rainbow public key results in a key recovery attack that is more efficient than previous attacks against Rainbow. Focusing on the parameter sets submitted to the final round of the NIST competition targeting SL I, III, and V, the estimated cost of the attack is $2^{127}, 2^{177}$, and 2^{226} gates, which is lower than the requirements set out by NIST (see Table 3.1) by a factor of $2^{16}, 2^{30}$, and 2^{46} respectively.

3.4 Conclusion

This chapter introduced security notions for digital signature schemes: UUF-KOA security means there is no efficient attacker algorithm that can forge signatures for arbitrary messages, given only the public key, while the more advanced EUF-CMA security property says there are no attacker algorithms that can output even just a single forgery for a message chosen by the attacker, even if the attacking algorithm is allowed to make arbitrarily many queries to a signing oracle.

Then, we summarized some new attacks I discovered against three post-quantum digital signature schemes. I found that the WalnutDSA scheme, which was submitted to the NIST PQC project, turned out to be completely breakable: the supposedly 128-bit secure parameters can be broken in less than a second and the 256-bit secure parameters could be broken in less than a minute. Unfortunately, Veridify, the corporation that owns WalnutDSA, is still pushing WalnutDSA to applications in the IoT and automotive industries, in part through partnerships with big players in the semiconductor industry such as STMicroelectronics and Intel. I also found that the LESS signature scheme was completely broken in practice, however with a new choice of parameters the scheme still has potential to be secure and efficient compared to other code-based signature schemes. Finally, I attacked Rainbow, one of the three finalist signature schemes in the NIST competition. Rainbow is one of the more mature MQ signature schemes, and hence it is not surprising that the break is less severe. The attacks are still too expensive to execute in practice with the resources available to me, but the attack is nonetheless more efficient than existing attacks by a wide margin (e.g., a factor 2^{20} for the NIST SL I parameters). Referring to my attacks on Rainbow, NIST requested community feedback on whether to standardize SPHINCS+ (one of the schemes chosen as alternative candidates) or to open up the NIST competition for a new round of signature schemes, suggesting that Rainbow will not be selected for standardization. As such, my research likely prevented a weak signature algorithm from being standardized and potentially achieving widespread adoption.

Chapter 4

Designing Secure and Efficient Signature Algorithms

There's a fool born every minute, and anyone who tries to write their own crypto algorithms definitely falls into this category.

Comment on "Schneier on Security" blogpost

If others had not been foolish, we should be so.

William Blake, *Proverbs of Hell*

In this chapter, we will finally talk about how to construct digital signature schemes. There are two commonly used approaches to construct signature schemes: the full-domain-hash approach (FDH), which converts a trapdoor function into a signature scheme, and the Fiat-Shamir approach that converts a zero-knowledge proof into a signature scheme. Nearly all the signature schemes we encountered in this thesis use one of these approaches. The trapdoor-based signature schemes include RSA (which uses modular exponentiation as trapdoor function), as well as Rainbow, WAVE, and Falcon. The zero-knowledge based signatures include ECC signature schemes, Dilithium[1], CSI-FiSh, and LESS.

[1]Dilithium uses the more general technique of Fiat-Shamir with aborts.

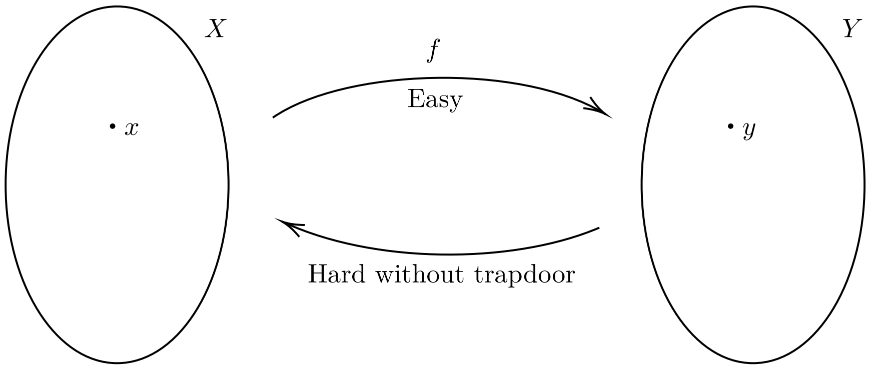


Figure 4.1: A trapdoor function $f : X \to Y$

After introducing these two paradigms with their advantages and disadvantages, I will briefly describe a number of signature schemes that I designed during my PhD research. The designs rely on existing hardness assumptions as much as possible, such that we can have some confidence in their security based on the lessons learned from cryptanalysing earlier algorithms. However, the new signatures improve over existing signatures in terms of speed, key size, and/or signature size, which makes them more suitable for practical applications.

4.1 Trapdoor-based signature algorithms

4.1.1 Trapdoor functions

A trapdoor function is a function f that is easy to compute (i.e., given x we can compute $y = f(x)$), but for which it is hard to compute preimages (i.e., given y, compute x such that $y = f(x)$), unless you have some secret information called the trapdoor (see Fig. 4.1). In the case of RSA signatures, the trapdoor functions are modular exponentiation maps:

$$f_{e,N} : \mathbb{Z}_N \to \mathbb{Z}_N : x \mapsto x^e \mod N\,,$$

where N is a product of two large primes p and q. These functions can be computed efficiently with the square-and-multiply algorithm, but it is believed to be hard (at least for classical computers) to compute x, given e, N, and $x^e \bmod N$. However, if you know the factorization of N, then one can efficiently compute x using the Chinese remainder theorem.

Where do trapdoor functions come from?

Often, a trapdoor function is constructed starting from some easy instance of a problem, and then somehow randomizing it to make it look like a random (and hopefully hard to solve) instance. We saw two examples of this in Chapter 2: in multivariate cryptography, one starts with some special multivariate map $\mathcal{F}$ for which one can efficiently sample preimages, and then it is randomized by composing it with linear maps $\mathcal{S}$ and $\mathcal{T}$ from both sides to get the actual trapdoor function $\mathcal{P} = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T}$. Similarly, the McEliece cryptosystem uses a code with a generator matrix G that can be decoded efficiently, and this is randomized by multiplying G with an invertible matrix S and a permutation matrix P to get the trapdoored code with generator matrix $G' = SGP$.

4.1.2 Full Domain Hash signatures

Given a family of surjective trapdoor functions, it is relatively straightforward to build a signature scheme with the Full Domain Hash approach (FDH), which uses a hash function $\mathcal{H}$ that maps messages into the range of the trapdoor functions.

Key Generation. The user generates a trapdoor function f, along with the trapdoor information that allows him to compute preimages. The public key is f, and the trapdoor information is the secret key.

Signing. To sign a message $m \in \{0,1\}^*$, the signer hashes the message to obtain a digest $h = \mathcal{H}(M)$. Then, using the trapdoor information, the signer generates a preimage for h, i.e., a value sig such that $f(\mathsf{sig}) = h$. This value sig is the signature.

Verification. To verify if a signature sig is valid for message m under public key f, the verifier checks if $f(\mathsf{sig}) = \mathcal{H}(m)$.

In the UUF-KOA game the adversary is given f, and a message m, and he needs to output a signature sig such that $f(\mathsf{sig}) = \mathcal{H}(m)$. Therefore, if we model the hash function as a function that for each message m outputs a random element $H(m)$ in the co-domain of f, then the signature is UUF-KOA secure if and only if the function f is hard to invert without knowledge of the trapdoor. However, EUF-CMA security (which we care about in practice), is not guaranteed from the assumption that f is a secure trapdoor function alone.

It is possible that by signing messages, the signer leaks information about the trapdoor, and that after seeing enough signed messages, an attacker has enough information to be able to forge signatures. Many signature schemes have fallen prey to this kind of attack (e.g., [53, 71]), so when designing or cryptanalysing a trapdoor-based signature scheme special care needs to be taken to ensure that signatures do not leak exploitable information about the secret key.

In some cases, it can be proven that the signatures do not leak information. For example, if the trapdoor function is a permutation[2], or more generally if the trapdoor is preimage sampleable (see [40]). The RSA trapdoor functions are permutations, so RSA signatures are not vulnerable to this kind of attack.

4.2 Zero-Knowledge-based signature algorithms

4.2.1 Zero-knowledge proofs of knowledge.

Zero-knowledge proofs are a wonderful tool, introduced by Goldwasser, Micali, and Rackoff in '89 [41], that allow someone to prove a statement is true, without revealing any information about why it is true. This seemingly impossible task can in fact be done using clever cryptographic protocols, for a wide variety of statements (ZK proofs exists for all languages in PSPACE if one-way functions exist). For the purpose of digital signatures, we are interested in zero-knowledge proofs *of knowledge*, where the prover convinces a verifier that he knows a solution to some computational problem, without revealing anything about the solution itself.

We want zero-knowledge proofs of knowledge to have three important properties: completeness, knowledge soundness and zero-knowledge. Completeness means that a prover can make the verifier accept the proof if he really knows a solution. Knowledge soundness means that if the protocol succeeds, then the verifier is convinced that the prover knows a solution to the computational problem. Zero-knowledge means that the verifier does not learn any information about the solution itself.

ZK proof for graph isomorphisms. A classical example is a zero-knowledge protocol to prove knowledge of an isomorphism between two graphs G_1 and G_2. A simple graph consists of a set of vertices V, and a set of edges E, which consists of unordered pairs of distinct elements in the vertex set V. Two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are isomorphic, if there exists a bijection $\phi : V_1 \to V_2$, such that $(v_1, v_2) \in E$ if and only if $(\phi(v_1), \phi(v_2)) \in E_2$ for all v_1, v_2 in V (see Fig 4.2 for an example).

[2]A function $f : X \to Y$ is a permutation if for every $y \in Y$ there exists a unique $x \in X$ such that $f(x) = y$.

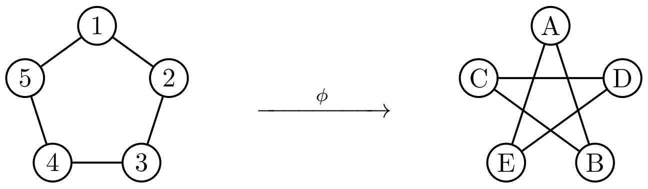


Figure 4.2: Example of two isomorphic graphs. An isomorphism is given by the map ϕ that sends 1 to A, 2 to B, and so on.

$$\begin{tikzcd}
 & G' \arrow[dr, dashed, "\phi \circ \rho^{-1}"] & \\
G_1 \arrow[ur, "\rho"] \arrow[rr, "\phi"] & & G_2
\end{tikzcd}$$

Figure 4.3: The classic zero-knowledge proof of knowledge of a graph isomorphism.

Suppose the prover P wants to convince a verifier V that he knows an isomorphism ϕ between two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$. To do this they can follow the following 3-round protocol.

- **Commitment phase.** The prover picks a random bijection ρ from V_1 to a new vertex set $\{1, \dots, l\}$ of size $l = |V_1| = |V_2|$. Then he computes a new set of edges

  $$E' = \{(\rho(x), \rho(y)) \,|\, \forall (x, y) \in E_1\} \ .$$

  He sends the new graph $G' = (V', E')$ to the verifier.

- **Challenge phase.** The verifier chooses a random challenge bit $c \in \{0, 1\}$ and sends it to the prover.

- **Response phase.** If $c = 0$ the prover sends ρ, otherwise the prover sends $\rho' = \phi \circ \rho^{-1}$.

- **Verification.** The verifier accepts the proof if $\rho(G_1) = G'$ in case $c = 0$, and he accepts if $\rho'(G') = G_2$ in case $c = 1$.

Completeness. It is clear that if the prover does know a valid graph isomorphism $\phi : G_1 \to G_2$, then if both parties follow the protocol honestly, then the verifier will accept the proof with probability 1.

Knowledge soundness. During the protocol, the prover sends a graph G' to the verifier. By doing this, the prover implicitly claims that he knows an isomorphism from G_1 to G' (namely ρ), and an isomorphism from G' to G_2 (namely $\rho' = \phi \circ \rho^{-1}$). If both claims are correct, then the prover must also know an isomorphism $\rho' \circ \rho = \phi$ from G_1 to G_2. The verifier is allowed to challenge one of the two claims, so if the prover was cheating, then he will be caught with a probability of at least 1/2. If the protocol is repeated k times, then the probability that a cheating prover manages to pass all the challenges is $(1/2)^k$, so by repeating the protocol a number of times, the verifier can be convinced that P knows an isomorphism except for an exponentially small probability. This property of the protocol is called knowledge soundness (with error[3] of 1/2).

Zero-Knowledge. Interestingly, the protocol does not reveal any information about the isomorphism ϕ. When the challenge bit is $b = 0$, the verifier gets to see the bijection ρ, and the graph G'. The bijection ρ was chosen uniformly at random, so this does not carry any information about ϕ, and G' can be computed from G_1 and ρ, so this does not contain any additional information. Similarly, in the case $b = 1$, the bijection ρ' is uniformly random, and G' can be computed from G_2 and ρ' so this case also reveals no information about ϕ. This property is called zero-knowledge.

Sigma protocols. The graph isomorphism proof of knowledge is an example of a sigma protocol, which is a class of zero-knowledge proofs of knowledge introduced by Cramer [21] with the following properties:

1. There are three rounds of communication: First the prover sends a commitment com, then the verifier sends a challenge c, which is chosen uniformly at random from some challenge space $\mathcal{C}$, and then the prover sends a response rsp.

2. Given accepting responses to two different challenges for the same commitment, it is possible to extract a solution to the problem. This property is called special soundness because it implies that the protocol is sound with error $1/|\mathcal{C}|$.

3. Anyone can create transcripts of executions of the protocol that are indistinguishable from real executions of the protocol. This property is called honest verifier zero-knowledge.

[3]The soundness error is the maximal probability with which a dishonest prover can convince the verifier.

Sigma protocols are very common in cryptography. We are interested in them because a sigma protocol for a hard problem can be transformed into a digital signature algorithm with the so-called Fiat-Shamir transform.

4.2.2 Fiat-Shamir Signatures.

If you have a sigma protocol for a hard problem, you can use it to authenticate yourself. You generate an instance of the problem, which you use as public key, and you keep the solution to yourself as secret key. Since the problem is hard, attackers will not be able to find a solution to your problem. Then, when you want to authenticate yourself, you execute the sigma protocol sufficiently many times to convince a verifier that you have a solution to your problem, which means that you must be authentic. Identification protocols are similar to signature schemes, but they are interactive, meaning that both parties need to be online at the same time, which makes them impractical for certain applications.

In ‘86, Fiat and Shamir found a method to get rid of the interaction, and convert a sigma protocol into a digital signature algorithm [35]. Interaction is only required to let the verifier choose the random challenges, so the solution is rather simple: instead of letting the verifier choose the random challenges, the signer determines the challenges himself, by hashing the public key pk, the commitments com, and the message m that needs signing with a cryptographic hash function $\mathcal{H}$. That is, the challenges are derived as $c^{(1)}, \ldots, c^{(k)} \leftarrow \mathcal{H}(\mathsf{pk}||\mathsf{com}||m)$ (k is the number of executions of the sigma protocol). The intuition is that the prover has no control over what the hash function outputs, so it should not make too much of a difference whether the challenges were chosen by the verifier or by the hash function. More formally, it can be proven that this transformation results in a EUF-CMA-secure signature scheme if the underlying problem is hard, and if we model the hash function $\mathcal{H}$ as a random oracle. Concretely, a Fiat-Shamir signature goes like this:

Key Generation. The user generates an instance of a hard problem, for which there is an efficient sigma protocol. The instance serves as the public key pk, the solution to the problem serves as the secret key sk.

Signing. The signer executes k independent executions of the first phase of the Sigma protocol to produce k commitments $\mathsf{com}^{(1)}, \ldots, \mathsf{com}^{(k)}$. Then he hashes the commitments with the public key and the message $H(\mathsf{pk}, \mathsf{com}, m)$ to obtain a list of challenges $c^{(1)}, \ldots, c^{(k)} \in \mathcal{C}^k$. Finally, the prover executes the response phase of the sigma protocol k times to produce responses $\mathsf{rsp}^{(1)}, \ldots, \mathsf{rsp}^{(k)}$. The signature is now $\mathsf{sig} = (\mathsf{com}^{(1)}, \ldots, \mathsf{com}^{(k)}, \mathsf{rsp}^{(1)}, \ldots, \mathsf{rsp}^{(k)})$.

Verification. The verifier hashes $H(\mathsf{pk}, \mathsf{com}, m)$ to obtain the same challenges

$c^{(1)}, \ldots, c^{(k)} \in \mathcal{C}^k$, and verifies for each i from 1 to k if $(\mathsf{com}^{(i)}, c^{(i)}, \mathsf{rsp}^{(i)})$ is a valid transcript of an execution of the sigma protocol using the verification phase of the sigma protocol.

4.3 Comparison of trapdoor-based and ZK-based signatures

Signature size. Since a trapdoor-based signature consists of only a single preimage for $\mathcal{H}(m)$, the signature size is typically small. For example, the Rainbow and Falcon signature schemes have a signature size of only 66 Bytes and 666 Bytes at NIST SL 1. In comparison, the size of ZK-based signatures is often much larger, because the signature is essentially a zero-knowledge proof, which can be complicated and large. For example, if the signature is based on a sigma protocol with soundness error 1/2, the base protocol needs to be repeated 128 times to reach a security level of 128 bits, which blows up the signature size by a factor 128.

Public key size. For ZK-based signatures the public key is some instance of a hard problem. Usually, most of this instance can be generated pseudo-randomly from a seed value. For example, in the case of Multivariate cryptography, the problem is to find a solution $\mathbf{s}$ such that $\mathcal{P}(\mathbf{s}) = \mathbf{t}$, for some multivariate quadratic map $\mathcal{P}$. In this case, the coefficients of $\mathcal{P}$ can be generated pseudo-randomly from a seed value seed. The user then picks a random $\mathbf{s}$ and computes $\mathbf{t} = \mathcal{P}(\mathbf{s})$. The public key is then represented by only seed and $\mathbf{t}$, which is very compact (e.g., 38 bytes for MUDFISH). Since public keys for trapdoor-based signature schemes need to hide some secret trapdoor structure, it is typically not possible to use this trick, which can make the public key very large (e.g., 158 KB for Rainbow and 3.2 MB for WAVE).

Provable security. ZK-based signatures are very amenable to provable security. One usually chooses a clean, random instance of an assumed hard problem as public key, and the knowledge soundness and ZK property of the zero-knowledge proof allow one to formally prove that breaking the security of the system is as hard as breaking that hard problem. In contrast, it is harder to give security proofs for trapdoor-based signature algorithms. Often, the trapdoor hides some structure, and it might not be possible to prove that this structure cannot be exploited, which leads to less natural hardness assumptions. For example, the RSA assumption[4] is less natural than the assumption that integer factorization

[4]The RSA assumption states that it is hard to find s given N, e, and $s^e \mod N$. In other words, the assumption is just that schoolbook RSA signatures are UUF-KOA secure.

is hard. Moreover, it is not always possible to prove that the signatures do not leak information about the secret key.

Contribution 3: Designing secure and efficient signature algorithms

This section deals with some of the signature schemes I designed during my PhD research. In particular, we will briefly discuss LUOV, a trapdoor-based multivariate signature scheme that I submitted to the NIST PQC project, CSI-FiSh a Fiat-Shamir signature scheme based on isogenies, and MUDFISH and SUSHSYFISH, two signature schemes that use a new framework of sigma protocols with helper, based on the permuted kernel problem and the MQ problem respectively.

C3.A: LUOV

As mentioned before, trapdoor-based multivariate signature schemes tend to have very large public keys, because a public key consists of a multivariate quadratic map $\mathcal{P} : \mathbb{F}_q^n \to \mathbb{F}_q^m$, which requires $\mathcal{O}(mn^2)$ coefficients to describe. There is a trade-off between the size of the finite field and the dimensions m and n. If q is small, we need to compensate with larger m and n to make the MQ problem hard enough, and vice versa. The optimal choice of q still results in public keys of more than 50 KB for UOV, one of the oldest and best understood MQ signature schemes.

The LUOV scheme, which stands for Lifted UOV, breaks this trade-off by simultaneously defining the public key $\mathcal{P}$ over a small finite field ($\mathbb{F}_2$) and choosing small dimensions n and m. Usually, this would make the problem of solving $\mathcal{P}(\mathbf{s}) = \mathbf{t}$ for $\mathbf{s}$ too easy, but the main idea is to define $\mathbf{t}$ over a large extension field, which forces the attack to look for $\mathbf{s}$ in an extension field, which makes the attack much more expensive. This, in combination with some other optimizations, reduced the public key size dramatically. At NIST SL I the public key size of LUOV was only 4.7 KB, which is an order of magnitude smaller than plain UOV. The details of the LUOV scheme are given in the paper of Chapter 13.

The central assumption for this to be secure was that solving a system $\mathcal{P}(\mathbf{s}) = \mathbf{t}$ for $\mathbf{s}$ with $\mathcal{P}$ defined over $\mathbb{F}_2$ and $\mathbf{t}$ defined over a large extension field $\mathbb{F}_{2^r}$ is equally difficult as solving a system where also $\mathcal{P}$ is defined over the extension field $\mathbb{F}_{2^r}$. Unfortunately, Ding *et al.* showed in two papers [32, 30] that this assumption is not valid for sufficiently underdetermined systems (i.e if n is sufficiently larger than m). This resulted in powerful attacks that were much more efficient than the NIST process required, and for one of the proposed parameter sets the attack could even be demonstrated in practice. As such, LUOV was not selected to proceed to the third round of the NIST project. However, the field lifting technique could still be used in the future to reduce the key size of MQ signature schemes. Quoting the NIST report: "The development of the aforementioned attacks shows that the lifting innovation is too new to be incorporated into a standard at this time; however, there is room for growth in this area. [...] LUOV has already inspired the application of field lifting for other schemes, and while it is premature to either trust or discard the lifting construction, NIST believes there is value in the development of the science in this direction." [4]

C3.B: CSI-FiSh

CSI-FiSh is a signature scheme I developed with Vercauteren and Kleinjung. It is based on a very simple sigma protocol that is very similar to the graph-isomorphism protocol we discussed earlier. In fact, the graph isomorphism protocol directly generalizes to any group action. For an action $\star$ of a finite group G on a set X, the generalized protocol allows to prove knowledge of a group element g such that $g \star x = y$ for given $x, y \in X$, in zero-knowledge.

There is a problem when trying to apply this protocol to the CSIDH group action (see Sect. 2.2.2) though. The acting group in question is the ideal class group $\mathcal{C}\ell(\mathcal{O})$ of an endomorphism ring $\mathcal{O}$ of a supersingular curve E_0. We know a set of generators of this group whose action we can evaluate efficiently, but the relations between these generators are hard to compute, which makes it hard to sample group elements at random and represent group elements uniquely, which is a requirement for the sigma protocol to work. De Feo and Galbraith worked around these problems with the Fiat-Shamir with aborts technique [22]. However, this comes at a cost in terms efficiency.

Our contribution was to compute the structure of the class group for a specific choice of parameters. This was a record-breaking computation: the previous record was the computation of a class group for an imaginary quadratic field with a 130-digit discriminant, while our computation corresponds to an imaginary quadratic field with a 154-digit discriminant. The computation took an estimated effort of 52 core-years. With the class group known, it becomes possible to sample elements of the class group uniformly at random and represent them uniquely. This makes it possible to use the simple graph-isomorphism-like sigma protocol. After some optimizations, this resulted in the first somewhat practical isogeny-based signature scheme, which we called CSI-FiSh (Commutative Supersingular Isogeny Fiat-Shamir). For 128 bits of classical security (which is currently the only security level available), a sign/verify operation takes 390 ms and the signature size is 263 bytes, which is faster than earlier isogeny-based signatures by a factor 300, and more compact by a factor 3.

In addition to efficient signatures, the computation of the class group structure has made it possible to construct more advanced cryptographic primitives from the CSIDH action, such as (linkable) ring signatures and threshold signatures [10, 24].

C3.C: MUDFISH/SUSHSYFISH

In this last section, I will introduce a framework of sigma protocols with helper that I developed in the paper of Chapter 14. These are the same as the sigma protocols from Sect. 4.2.1, except that in addition to the prover and the verifier, there is an additional trusted helper party, that does a setup at the beginning of each execution of the protocol and then goes to sleep for the remainder of the protocol. Trusted parties are hard to come by, so sigma protocols with helper might not seem very useful. However, there is a transformation to remove the helper and transform the sigma protocol with helper into a standard sigma protocol, which can in turn be transformed into a signature scheme with the Fiat-Shamir transform. It turns out that first constructing a sigma protocol with helper, and then converting it into a standard sigma protocol can result in more efficient protocols, compared to existing ZK-proofs. For example, we construct new sigma protocols for the MQ problem and the permuted kernel problem, which have a soundness error of $1/q$, which is much better than the existing protocols which have a soundness error of $(1+q)/2q \approx 1/2$. This means that the base protocol needs to be repeated fewer times, which makes the overall protocol more efficient. By applying the Fiat-Shamir transform to the new sigma protocol we also obtain more efficient signature schemes.

MUDFISH. The new signature scheme based on the sigma protocol for the MQ problem is called the MUltivariate quaDratic FIat-SHamir scheme (MUDFISH). MUDFISH is EUF-CMA secure in the quantum random oracle model, assuming the hardness of the MQ problem. Compared to MQDSS, the signature scheme in the NIST competition which is based on the exact same assumption, MUDFISH is more efficient: the MUDFISH signature size of 14 KB, which is a factor 2 improvement over the signature size of MQDSS, and our implementation of MUDFISH is twice as fast as the optimized implementation of MQDSS that was submitted to the NIST PQC project.

SUSHSYFISH. The signature scheme based on the PKP proof is called the ShUffled Solution to Homogeneous linear SYstem FIat-SHamir (SUSHSYFISH). This scheme is EUF-CMA secure in the QROM assuming the permuted kernel problem is hard. SUSHSYFISH was a significant improvement over PKP-DSS, a signature scheme based on the same hardness assumption (see Sect 2.2.5). However, since then an optimized variant of PKP-DSS appeared on ePrint [59], whose signature size if only slightly larger than that of SUSHSYFISH (e.g., 13 KB versus 12 KB at NIST SL I), but which should be significantly faster (even though no implementation is available yet). This new variant of PKP-DSS would likely be a better choice compared to SUSHSYFISH for most applications.

4.4 Conclusion

This chapter discussed and compared the two most prominent strategies for constructing digital signature schemes: trapdoor-based schemes, and zero-knowledge-based schemes. Then, we introduced some of the schemes I designed during my PhD research. The intention behind these schemes was to use existing hardness assumptions and to build signature schemes that are more efficient from these assumptions. The exception was LUOV, a trapdoor-based scheme which achieved a very significant improvement in public key size over existing MQ schemes, but at the cost of a new hardness assumption related to systems of multivariate quadratic polynomials with certain coefficients restricted to a subfield. This assumption was cryptanalysed by Ding *et al.*, resulting in more powerful attacks. The other schemes introduced in this chapter CSI-FiSh, MUDFISH, and SUSHSYFISH, are based on zero-knowledge proofs, and have security reductions from more established hardness assumptions. These new schemes made significant performance improvements over existing schemes that relied on the same hardness assumptions.

Chapter 5

Conclusion

Harder, Better, Faster, Stronger.
Our work is never over.

– Daft Punk

Given the fast-paced progress in quantum computing technology, it is necessary to design, standardize and deploy post-quantum digital signatures to protect our IT infrastructure from attackers with access to quantum computers. This thesis discussed three important aspects of post-quantum digital signatures. Chapter 2 introduced some hard computational problems that post-quantum signatures can be built from, including some less common assumptions that I worked on. Chapter 3 deals with the cryptanalysis of these assumptions and the systems built on them, including some definitions of what it means for a digital signature scheme to be (in)secure. In particular, we discuss some of the vulnerabilities that I discovered in existing post-quantum signature schemes. Chapter 4 discusses the two most common strategies to construct digital signature algorithms, and my efforts to make these algorithms more efficient. In this chapter, we summarize some conclusions and contributions regarding each of these aspects, and we discuss some directions for future research.

Objective 1: Diversifying hardness assumptions

Post-quantum digital signature schemes can be built from a number of hardness assumptions. However, we have yet to find a silver bullet. The existing assumptions lead to signature schemes that are less efficient than existing pre-quantum signature schemes, and their security is often much less understood. Therefore, it is important to diversify the set of hardness assumptions by looking at new problems, in the hope that these could lead to secure and more efficient signature schemes in the long term.

I contributed to this objective by studying two relatively unknown hardness assumptions: the hardness of the permuted kernel problem and the security of the Legendre PRF. On one hand, my research showed that reasonably efficient signatures algorithms can be built based on these hardness assumptions. One of the signature schemes based on the permuted kernel problem won a third prize in the Chinese post-quantum cryptography competition. On the other hand, I improved our understanding of the security of these schemes by cryptanalysing the Legendre PRF. Even though some more efficient attacks are found, these attacks did not threaten the new signature scheme. More recent quantum analysis of the security of the Legendre PRF seems to suggest that the Legendre PRF holds up well against quantum cryptanalysis [37].

Future work. Firstly, more (quantum) cryptanalysis of the Permuted Kernel Problem and the Legendre PRF is necessary to gain more confidence in the security of signature schemes based on these problems. Secondly, it would be interesting to construct more primitives or more efficient signature schemes from these new assumptions. Independent researchers have already optimized the PKP-DSS scheme, resulting in significant performance improvements [59]. PorcRoast seems to be running into the limitations of the MPC-in-the-head framework, since only 1/3 of the signature size is related to the verification of the power residue symbols, and the rest is overhead from the MPC-in-the-head construction (this overhead mostly consists of seed and commitment values). Therefore, it would be interesting to construct zero-knowledge proofs outside the MPC-in-the-head framework to make even more efficient signature schemes based on power residue symbols.

Objective 2: Cryptanalysis

While the security of hash-based signatures is well understood and robust, the security analysis of the other families of post-quantum cryptography is less stable. For example, on the front of lattice-based cryptography, there is steady progress in lattice reduction algorithms (both in theory and practice), and new results regarding the impact of algebraic structure of lattices on security are found regularly. In the other branches of post-quantum signatures, the situation is also unstable.

This thesis presented three attacks against post-quantum digital signature schemes. Two of the attacks are powerful enough to break schemes claiming

128 bits of security in practice: I found that the WalnutDSA algorithm, which was submitted to the NIST PQC project and which is being used in practice, could be broken in under a second. Similarly, using a new technique to find out if two error-correcting codes are equivalent or not, the LESS signature scheme could also be broken in mere seconds. This new technique will help assess the security of future signature algorithms based on the code equivalence problem.

I also improved the cryptanalysis of UOV and Rainbow, two of the oldest and most studied signature schemes in multivariate quadratic cryptography. Rainbow is one of the three remaining finalist signature schemes in the NIST PQC project. My new attacks are more efficient than the existing attacks by a factor $2^{20}, 2^{40}$ and 2^{55} for the NIST security levels I, III, and V respectively. In order to protect against the attacks, Rainbow needs significantly larger parameters, which eliminates the small performance advantage that Rainbow had over UOV. The response of NIST following my attack on Rainbow seems to indicate that Rainbow will not be selected for standardisation. As such, my research might have prevented a weak cryptosystem from being standardized and being used by millions of users.

Future work. All the branches of post-quantum cryptography could benefit from more cryptanalysis, but one particularly interesting target for cryptanalysis is the MinRank problem. This problem appears in attacks against Multivariate-based and code-based cryptosystems, so a more thorough understanding of this problem is required to have confidence in the security of these cryptosystems. A recent breakthrough result [7] drastically improved algorithms for solving this problem, so further improvements might be within reach. A different direction is the cryptanalysis of the SQISign scheme. This isogeny-based signature scheme has remarkably short key and signature sizes. However, the scheme is based on rather complex mathematical assumptions, so independent cryptanalysis is needed.

Objective 3: Design of efficient signature schemes

An important objective is to make post-quantum signature schemes faster and the keys and signatures more compact. This will make the necessary transition to post-quantum cryptography go smoother, which means a larger share of applications will be protected when the first quantum computers capable of breaking RSA and ECC are constructed.

This thesis presented a number of signature schemes I (co)-designed that are more efficient than other signature schemes based on the same hardness assumptions. A striking example is CSI-FiSh, a new isogeny-based signature scheme that is faster than earlier isogeny-based signatures by a factor 300, and more compact by a factor 3. A second example is MUDFISH, which is a factor 2 faster and a factor 2 smaller than MQDSS, a digital signature scheme based only on the hardness of the MQ problem. Despite substantial progress, there is still a significant gap in the performance of post-quantum digital signatures and the pre-quantum alternatives they have to replace.

Future work. Post-quantum cryptography is still in its infancy, so there are still plenty of opportunities to make faster, more secure, and more compact signature schemes. I will only mention a few directions that I aspire to work on in the future. Since we have high confidence in the security of hash-based signatures, performance improvements for these algorithms would be of much practical interest. Significant improvements can be achieved by using an optimal encoding function in the Winternitz OTS building block, and by doing a more thorough parameter search. A different direction is multivariate quadratic signatures. I believe the new insights on the UOV trapdoor that I obtained from my cryptanalysis work, can help build more secure and more efficient multivariate-quadratic signature schemes. Moreover, it should be possible to prove the EUF-CMA security of the new signature scheme, based on natural mathematical problems.

References

The most important function of a bibliographic entry is to help the reader obtain a copy of the cited work.

— Daniel J. Bernstein

[1] Digital Signature Standard (DSS). National Institute of Standards and Technology (NIST), FIPS PUB 186-4, U.S. Department of Commerce, July 2013.

[2] Ajtai, M. Generating hard instances of lattice problems (extended abstract). In *28th ACM STOC* (May 1996), ACM Press, pp. 99–108.

[3] Alagic, G., Alagic, G., Alperin-Sheriff, J., Apon, D., Cooper, D., Dang, Q., Liu, Y.-K., Miller, C., Moody, D., Peralta, R., et al. Status report on the first round of the NIST post-quantum cryptography standardization process, 2019. `https://csrc.nist.gov/publications/detail/nistir/8240/final`.

[4] Alagic, G., Alperin-Sheriff, J., Apon, D., Cooper, D., Dang, Q., Kelsey, J., Liu, Y.-K., Miller, C., Moody, D., Peralta, R., et al. Status report on the second round of the NIST post-quantum cryptography standardization process, 2020. `https://csrc.nist.gov/publications/detail/nistir/8309/final`.

[5] Atkins, D., Anshel, I., Goldfeld, D., and Gunnells, P. E. WalnutDSA. Tech. rep., National Institute of Standards and Technology, 2017. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions`.

[6] Bahr, F., Boehm, M., Franke, J., and Kleinjung, T. Factorization of RSA-200, 2005. https://members.loria.fr/PZimmermann/records/rsa200.

[7] Bardet, M., Bros, M., Cabarcas, D., Gaborit, P., Perlner, R. A., Smith-Tone, D., Tillich, J.-P., and Verbel, J. A. Improvements of algebraic attacks for solving the rank decoding and MinRank problems. In *ASIACRYPT 2020, Part I* (Dec. 2020), S. Moriai and H. Wang, Eds., vol. 12491 of *LNCS*, Springer, Heidelberg, pp. 507–536.

[8] Beullens, W. Improved cryptanalysis of UOV and Rainbow. Cryptology ePrint Archive, Report 2020/1343, 2020. `https://eprint.iacr.org/2020/1343`.

[9] Beullens, W. Sigma protocols for MQ, PKP and SIS, and Fishy signature schemes. In *EUROCRYPT 2020, Part III* (May 2020), A. Canteaut and Y. Ishai, Eds., vol. 12107 of *LNCS*, Springer, Heidelberg, pp. 183–211.

[10] Beullens, W., Katsumata, S., and Pintore, F. Calamari and Falafl: Logarithmic (linkable) ring signatures from isogenies and lattices. In *ASIACRYPT 2020, Part II* (Dec. 2020), S. Moriai and H. Wang, Eds., vol. 12492 of *LNCS*, Springer, Heidelberg, pp. 464–492.

[11] Beullens, W., Kleinjung, T., and Vercauteren, F. CSI-FiSh: Efficient isogeny based signatures through class group computations. In *ASIACRYPT 2019, Part I* (Dec. 2019), S. D. Galbraith and S. Moriai, Eds., vol. 11921 of *LNCS*, Springer, Heidelberg, pp. 227–247.

[12] Biasse, J.-F., Micheli, G., Persichetti, E., and Santini, P. LESS is more: Code-based signatures without syndromes. In *AFRICACRYPT 20* (July 2020), A. Nitaj and A. M. Youssef, Eds., vol. 12174 of *LNCS*, Springer, Heidelberg, pp. 45–65.

[13] Boudot, F., Gaudry, P., Guillevic, A., Heninger, N., Thomé, E., and Zimmermann, P. Comparing the difficulty of factorization and discrete logarithm: A 240-digit experiment. In *CRYPTO 2020, Part II* (Aug. 2020), D. Micciancio and T. Ristenpart, Eds., vol. 12171 of *LNCS*, Springer, Heidelberg, pp. 62–91.

[14] Casanova, A., Faugère, J.-C., Macario-Rat, G., Patarin, J., Perret, L., and Ryckeghem, J. GeMSS. Tech. rep., National Institute of Standards and Technology, 2020. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions`.

[15] Castryck, W., Lange, T., Martindale, C., Panny, L., and Renes, J. CSIDH: An efficient post-quantum commutative group action. In *ASIACRYPT 2018, Part III* (Dec. 2018), T. Peyrin and S. Galbraith, Eds., vol. 11274 of *LNCS*, Springer, Heidelberg, pp. 395–427.

[16] Chen, M.-S., Hülsing, A., Rijneveld, J., Samardjiska, S., and Schwabe, P. SOFIA: $\mathcal{MQ}$-based signatures in the QROM. In *PKC 2018, Part II* (Mar. 2018), M. Abdalla and R. Dahab, Eds., vol. 10770 of *LNCS*, Springer, Heidelberg, pp. 3–33.

[17] Chuengsatiansup, C., Prest, T., Stehlé, D., Wallet, A., and Xagawa, K. ModFalcon: Compact signatures based on module-NTRU lattices. In *ASIACCS 20* (Oct. 2020), H.-M. Sun, S.-P. Shieh, G. Gu, and G. Ateniese, Eds., ACM Press, pp. 853–866.

[18] Cook, S. The P vs. NP problem. CLAY mathematics foundation millenium problems. `https://www.claymath.org/millennium-problems/p-vs-np-problem`.

[19] Courtois, N., Klimov, A., Patarin, J., and Shamir, A. Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In *EUROCRYPT 2000* (May 2000), B. Preneel, Ed., vol. 1807 of *LNCS*, Springer, Heidelberg, pp. 392–407.

[20] Couveignes, J.-M. Hard homogeneous spaces. Cryptology ePrint Archive, Report 2006/291, 2006. `http://eprint.iacr.org/2006/291`.

[21] Cramer, R. *Modular Design of Secure yet Practical Cryptographic Protocols.* PhD thesis, Jan. 1997.

[22] De Feo, L., and Galbraith, S. D. SeaSign: Compact isogeny signatures from class group actions. In *EUROCRYPT 2019, Part III* (May 2019), Y. Ishai and V. Rijmen, Eds., vol. 11478 of *LNCS*, Springer, Heidelberg, pp. 759–789.

[23] De Feo, L., Kohel, D., Leroux, A., Petit, C., and Wesolowski, B. SQISign: Compact post-quantum signatures from quaternions and isogenies. In *ASIACRYPT 2020, Part I* (Dec. 2020), S. Moriai and H. Wang, Eds., vol. 12491 of *LNCS*, Springer, Heidelberg, pp. 64–93.

[24] De Feo, L., and Meyer, M. Threshold schemes from isogeny assumptions. In *PKC 2020, Part II* (May 2020), A. Kiayias, M. Kohlweiss, P. Wallden, and V. Zikas, Eds., vol. 12111 of *LNCS*, Springer, Heidelberg, pp. 187–212.

[25] Debris-Alazard, T., Sendrier, N., and Tillich, J.-P. Wave: A new family of trapdoor one-way preimage sampleable functions based on codes. In *ASIACRYPT 2019, Part I* (Dec. 2019), S. D. Galbraith and S. Moriai, Eds., vol. 11921 of *LNCS*, Springer, Heidelberg, pp. 21–51.

[26] Deutsch, D., and Jozsa, R. Rapid solution of problems by quantum computation. *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences 439*, 1907 (1992), 553–558.

[27] Diffie, W., and Hellman, M. E. New directions in cryptography. *IEEE Transactions on Information Theory 22*, 6 (1976), 644–654.

[28] Ding, C. T. A. P. J. Improved key recovery of the HFEv- signature scheme. Cryptology ePrint Archive, Report 2020/1424, 2020. `https://eprint.iacr.org/2020/1424`.

[29] Ding, J., Chen, M.-S., Petzoldt, A., Schmidt, D., Yang, B.-Y., Kannwischer, M., and Patarin, J. Rainbow. Tech. rep., National Institute of Standards and Technology, 2020. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions`.

[30] Ding, J., Deaton, J., Schmidt, K., Zhang, Z., et al. Cryptanalysis of the lifted unbalanced oil vinegar signature scheme. In *Annual International Cryptology Conference* (2020), Springer, pp. 279–298.

[31] Ding, J., and Schmidt, D. Rainbow, a new multivariable polynomial signature scheme. In *ACNS 05* (June 2005), J. Ioannidis, A. Keromytis, and M. Yung, Eds., vol. 3531 of *LNCS*, Springer, Heidelberg, pp. 164–175.

[32] Ding, J., Zhang, Z., Deaton, J., Schmidt, K., and Vishakha, F. New attacks on lifted unbalanced oil vinegar. In *the 2nd NIST PQC Standardization Conference* (2019).

[33] Dworkin, M. J. Sha-3 standard: Permutation-based hash and extendable-output functions. Tech. rep., 2015. `https://www.nist.gov/publications/sha-3-standard-permutation-based-hash-and-extendable-output-functions`.

[34] Faugere, J.-C. A new efficient algorithm for computing Gröbner bases (F4). *Journal of pure and applied algebra 139*, 1-3 (1999), 61–88.

[35] Fiat, A., and Shamir, A. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO'86* (Aug. 1987), A. M. Odlyzko, Ed., vol. 263 of *LNCS*, Springer, Heidelberg, pp. 186–194.

[36] Florit, E., and Finol, G. Isogeny graphs of supersingular elliptic curves, Nov. 2020.

[37] Frixons, P., and Schrottenloher, A. Quantum security of the legendre prf. Tech. rep., Cryptology ePrint Archive, Report 2021/149, 2021. https://eprint. iacr. org ....

[38] Garey, M. R. Computers and intractability: A guide to the theory of np-completeness. *Revista Da Escola De Enfermagem Da USP 44*, 2 (1979), 340.

[39] Gentry, C. Fully homomorphic encryption using ideal lattices. In *41st ACM STOC* (May / June 2009), M. Mitzenmacher, Ed., ACM Press, pp. 169–178.

[40] Gentry, C., Peikert, C., and Vaikuntanathan, V. Trapdoors for hard lattices and new cryptographic constructions. In *40th ACM STOC* (May 2008), R. E. Ladner and C. Dwork, Eds., ACM Press, pp. 197–206.

[41] Goldwasser, S., Micali, S., and Rackoff, C. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing 18*, 1 (1989), 186–208.

[42] Harvey, D., and Van Der Hoeven, J. On the complexity of integer matrix multiplication. working paper or preprint, Oct. 2014.

[43] Jao, D., and De Feo, L. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In *Post-Quantum Cryptography - 4th International Workshop, PQCrypto 2011* (Nov. / Dec. 2011), B.-Y. Yang, Ed., Springer, Heidelberg, pp. 19–34.

[44] Joux, A., and Vitse, V. A crossbred algorithm for solving Boolean polynomial systems. Cryptology ePrint Archive, Report 2017/372, 2017. `http://eprint.iacr.org/2017/372`.

[45] Kocher, P. C. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *CRYPTO'96* (Aug. 1996), N. Koblitz, Ed., vol. 1109 of *LNCS*, Springer, Heidelberg, pp. 104–113.

[46] Kocher, P. C., Jaffe, J., and Jun, B. Differential power analysis. In *CRYPTO'99* (Aug. 1999), M. J. Wiener, Ed., vol. 1666 of *LNCS*, Springer, Heidelberg, pp. 388–397.

[47] Leech, D. P., Ferris, S., Scott, J. T., et al. The economic impacts of the advanced encryption standard, 1996–2017. *Annals of Science and Technology Policy 3*, 2 (2019), 142–257.

[48] Lyubashevsky, V., Ducas, L., Kiltz, E., Lepoint, T., Schwabe, P., Seiler, G., Stehlé, D., and Bai, S. CRYSTALS-DILITHIUM. Tech. rep., National Institute of Standards and Technology, 2020. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions`.

[49] Matsumoto, T., and Imai, H. Public quadratic polynominal-tuples for efficient signature-verification and message-encryption. In *EUROCRYPT'88* (May 1988), C. G. Günther, Ed., vol. 330 of *LNCS*, Springer, Heidelberg, pp. 419–453.

[50] McEliece, R. J. A public-key cryptosystem based on algebraic coding theory. DSN Progress Report 42-44, Jet Propulsion Laboratories, Pasadena, 1978.

[51] Mestre, J.-F. La méthode des graphes. exemples et applications. In *Proceedings of the international conference on class numbers and fundamental units of algebraic number fields (Katata)* (1986), Citeseer, pp. 217–242.

[52] Mosca, M., and Piani, M. Quantum Threat Timeline Report, 2019. https://globalriskinstitute.org/publications/quantum-threat-timeline/.

[53] Nguyen, P. Q., and Regev, O. Learning a parallelepiped: Cryptanalysis of GGH and NTRU signatures. In *EUROCRYPT 2006* (May / June 2006), S. Vaudenay, Ed., vol. 4004 of *LNCS*, Springer, Heidelberg, pp. 271–288.

[54] NIST. NIST PQC email list, 2016. https://groups.google.com/a/list.nist.gov/forum/#!forum/pqc-forum.

[55] NIST. Submission requirements and evaluation criteria for the post-quantum cryptography standardization process, 2016. https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/Call-for-Proposals.

[56] Patarin, J. Cryptanalysis of the Matsumoto and Imai public key scheme of Eurocrypt'88. In *Annual International Cryptology Conference* (1995), Springer, pp. 248–261.

[57] Patarin, J. The oil and vinegar signature scheme. In *Dagstuhl Workshop on Cryptography September, 1997* (1997).

[58] Prest, T., Fouque, P.-A., Hoffstein, J., Kirchner, P., Lyubashevsky, V., Pornin, T., Ricosset, T., Seiler, G., Whyte, W., and Zhang, Z. FALCON. Tech. rep., National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions.

[59] Ransom, R. Constant-time verification for cut-and-choose-based signatures. Cryptology ePrint Archive, Report 2020/1184, 2020. https://eprint.iacr.org/2020/1184.

[60] Rivest, R. L., Shamir, A., and Adleman, L. M. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the Association for Computing Machinery 21*, 2 (1978), 120–126.

[61] Rostovtsev, A., and Stolbunov, A. Public-Key Cryptosystem Based On Isogenies. Cryptology ePrint Archive, Report 2006/145, 2006. http://eprint.iacr.org/2006/145.

[62] Samardjiska, S., Chen, M.-S., Hulsing, A., Rijneveld, J., and Schwabe, P. MQDSS. Tech. rep., National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions.

[63] Schnorr, C.-P. Efficient identification and signatures for smart cards. In *CRYPTO'89* (Aug. 1990), G. Brassard, Ed., vol. 435 of *LNCS*, Springer, Heidelberg, pp. 239–252.

[64] Shamir, A. An efficient identification scheme based on permuted kernels (extended abstract) (rump session). In *CRYPTO'89* (Aug. 1990), G. Brassard, Ed., vol. 435 of *LNCS*, Springer, Heidelberg, pp. 606–609.

[65] Shor, P. W. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science* (1994), Ieee, pp. 124–134.

[66] Silverman, J. H. *The arithmetic of elliptic curves*, vol. 106. Springer Science & Business Media, 2009.

[67] Simon, D. R. On the power of quantum cryptography. In *35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, USA* (1994), pp. 116–123.

[68] Stern, J. A new identification scheme based on syndrome decoding. In *CRYPTO'93* (Aug. 1994), D. R. Stinson, Ed., vol. 773 of *LNCS*, Springer, Heidelberg, pp. 13–21.

[69] Thomae, E., and Wolf, C. Solving underdetermined systems of multivariate quadratic equations revisited. In *PKC 2012* (May 2012), M. Fischlin, J. Buchmann, and M. Manulis, Eds., vol. 7293 of *LNCS*, Springer, Heidelberg, pp. 156–171.

[70] Yoo, Y., Azarderakhsh, R., Jalali, A., Jao, D., and Soukharev, V. A post-quantum digital signature scheme based on supersingular isogenies. In *FC 2017* (Apr. 2017), A. Kiayias, Ed., vol. 10322 of *LNCS*, Springer, Heidelberg, pp. 163–181.

[71] Yu, Y., and Ducas, L. Learning strikes again: The case of the DRS signature scheme. In *ASIACRYPT 2018, Part II* (Dec. 2018), T. Peyrin and S. Galbraith, Eds., vol. 11273 of *LNCS*, Springer, Heidelberg, pp. 525–543.

Part II

Publications

Publications

The following nine chapters contain a selection of my research papers. Some of my papers were not included in this thesis, because they were not related to post-quantum signatures, or simply in the interest of space. These papers are listed here:

Other Papers

Beullens W., Katsumata S., Pintore F. Calamari and Falafl: Logarithmic (Linkable) Ring Signatures from Isogenies and Lattices. In: Advances in Cryptology – ASIACRYPT 2020. LNCS, vol 12492, pages 464-492. Springer, 2020.

Beullens W., Wee H. Obfuscating Simple Functionalities from Knowledge Assumptions. In: Public-Key Cryptography – PKC 2019. LNCS, vol 11443, pages 254-283. Springer, 2019.

Beullens W., Preneel B., Szepieniec A. Public Key Compression for Constrained Linear Signature Schemes. In: Selected Areas in Cryptography – SAC 2018. LNCS, vol 11349, pages 300-321. Springer, 2019.

Szepieniec A., Beullens W., Preneel B. MQ Signatures for PKI. In: Post-Quantum Cryptography. PQCrypto 2017. LNCS, vol 10346, pages 224-240. Springer, 2017.

Beullens W., Disson L., Pedersen R., Vercauteren F. CSI-RAShi: Distributed key generation for CSIDH. IACR ePrint report 2020/1323. Accepted to PQCRYPTO 2021, proceedings not yet published at the time of writing.

Chapter 6

PKP-DSS

Life truly begins only after you have put your things in order.

–Marie Kondo, Spark Joy, 2016

Publication Data

Beullens W., Faugère JC., Koussa E., Macario-Rat G., Patarin J., Perret L. PKP-Based Signature Scheme. In: Progress in Cryptology – INDOCRYPT 2019. Lecture Notes in Computer Science, vol 11898, pages pp 3-22. Springer, 2019.

My Contribution

One of the main authors. I optimized the signature scheme and wrote security proofs. I wrote the implementation and did the benchmarking.

PKP-Based Signature Scheme

Ward BEULLENS[1], Jean-Charles FAUGÈRE[2], Eliane KOUSSA[3], Gilles MACARIO-RAT[4], Jacques PATARIN[5], and Ludovic PERRET[2]

[1] imec-COSIC, KU Leuven ward.beullens@esat.kuleuven.be
[2] INRIA and Sorbonne Universities/UPMC Uni Paris 6
jean-charles.faugere@inria.fr, ludovic.perret@lip6.fr
[3] Versailles Laboratory of Mathematics, UVSQ EJKoussa@outlook.com
[4] Orange gilles.macariorat@orange.com
[5] Versailles Laboratory of Mathematics, UVSQ, CNRS, University of Paris-Saclay,
jpatarin@club-internet.fr

Abstract. In this document, we introduce PKP-DSS: a Digital Signature Scheme based on the Permuted Kernel Problem(PKP) [23]. PKP is a simple NP-hard [10] combinatorial problem that consists of finding a kernel for a publicly known matrix, such that the kernel vector is a permutation of a publicly known vector. This problem was used to develop an Identification Scheme (IDS) which has a very efficient implementation on low-cost smart cards. From this zero-knowledge identification scheme, we derive PKP-DSS with the traditional Fiat-Shamir transform [9]. Thus, PKP-DSS has a security that can be provably reduced, in the *(classical) random oracle model*, to the hardness of random instances of PKP (or, if wanted, to any specific family of PKP instances). We propose parameter sets following the thorough analysis of the State-of-the-art attacks on PKP presented in [**17**]. We show that PKP-DSS is competitive with other signatures derived from Zero-Knowledge identification schemes. In particular, PKP-DSS-128 gives a signature size of approximately 20 KBytes for 128 bits of classical security, which is approximately 30% smaller than MQDSS. Moreover, our proof-of-concept implementation shows that PKP-DSS-128 is an order of magnitude faster than MQDSS which in its turn is faster than Picnic2, SPHINCS,...
Since the PKP is NP-hard and since there are no known quantum attacks for solving PKP significantly better than classical attacks, we believe that our scheme is post-quantum secure.

Keywords: public-key cryptography, Fiat-Shamir, post-quantum cryptography, 5-pass identification scheme, Public-key Signature, Permuted Kernel Problem.

1 Introduction

The construction of large quantum computers would break most public-key cryptographic schemes in use today because they rely on the discrete logarithm problem or the integer factorization problem. Even though it isn't clear when large scale quantum computation would be feasible, it is important to anticipate quantum computing and design new public key cryptosystems that are resistant to quantum attacks. Therefore, there currently is a large research effort to develop new post-quantum secure schemes, and a Post-Quantum Cryptography standardization process has been initiated by the American National Institute of Standards and Technology (`https://www.nist.gov/`). Because of this, there has been renewed interest in constructing signature schemes by applying the Fiat-Shamir transform [9] to Zero-Knowledge Identification Schemes. In particular, we are interested in post-quantum cryptographic schemes whose security relies on the quantum hardness of some NP-Hard problem [2]. One of those problems is the Permuted Kernel Problem: the problem of finding a permutation of a known vector such that the resulting vector is in the kernel of a given matrix. This is a classical NP-Hard combinatorial problem which requires only simple operations such as basic linear algebra and permuting the entries of a vector. For quite some time, no new attacks on PKP have been discovered, which makes it possible to confidently estimate the concrete hardness of the problem.

In 1989, Shamir [23] introduced a five-pass ZK-Identification scheme, based on the PKP. This work uses the Fiat-Shamir transform [9] on this identification scheme to develop a signature scheme that is provably secure in the Random Oracle Model (ROM). However, since our goal is to have a post-quantum scheme, we should also consider attackers in the Quantum Random Oracle Model (QROM). The security of the Fiat-Shamir transform in the QROM has been studied in [27,26,25], where the authors of [27,25] explain that the Fiat-Shamir transform might not be secure against quantum computers. Thus, new techniques with extra properties (such as "lossy IDS") were developed to obtain a quantum-secure transform. However, more recently, a number of works have proven the Fiat-Shamir construction secure in the QROM[26,13] under very mild conditions. So far, none of these works apply to five-round protocols (which is the kind of protocol we are considering in this work), but it is conceivable that the results can be generalized to five-pass protocols, including ours. We consider this an important open problem in post-quantum cryptography.

Previous work and State-of-the-art. Since quantum computers are expected not to be capable of solving *NP*-Hard problems in sub-exponential time (in worst case), Zero-knowledge Identification schemes based on such problems are interesting candidates for Post-Quantum Cryptography. The Fiat-Shamir transform [9] is a technique that can convert such a zero-knowledge authentication scheme into a signature scheme. This approach was taken by Chen et al. [7], who applied the Fiat-Shamir transform to a 5-pass identification scheme of Sakumoto et al. [22]. This identification scheme relies on the hardness of the (NP-Hard) problem of finding a solution to a set of multivariate quadratic equations. Chen et al. proved that, in the random oracle model, applying the Fiat-Shamir transform to this 5-pass identification scheme results in a secure signature scheme. A concrete parameter choice and an efficient implementation of this signature scheme (which is called MQDSS) were developed, and this was one of the submissions to the NIST PQC standardization project. At a security level of 128 bits, the MQDSS scheme comes with a public key of 46 Bytes, a secret key of 16 Bytes and a signature size of approximately 28 Kilobytes.

A different line of work resulted in the Picnic signature scheme. Chase et al. [6] constructed this digital signature scheme by applying the Fiat-Shamir transform to an identification scheme whose security relies purely on symmetric primitives. At the 128-bit security level Picnic has a public key of 32 Bytes, a secret key of 16 Bytes and signatures of approximately 33 Kilobytes. There is a second version of this signature scheme, where the signatures are only 13.5 Kilobytes, but Picnic2 is 45 times slower than the original Picnic for signing and 25 times slower for verification.

Main results. The main contribution of this paper is to present PKP-DSS, a new post-quantum secure signature scheme. Similar to the approaches cited above, we use the Fiat-Shamir transform to construct a signature scheme from the 5-pass PKP identification scheme by Shamir [23]. Following the complexity analysis of the PKP [17], we choose secure parameter sets of the signature scheme for 128/192/256 of classical security level. To date, there are no known quantum algorithms for solving PKP (other than combining Grover search with the classical algorithms), so we claim that our signatures achieve the NIST security levels I/III and V respectively. However, we recognize that the (quantum) hardness of PKP deserves more research, and we hope that this work will inspire researchers to investigate this topic further.

We have developed a constant-time C implementation of the new signature scheme. By constant-time we mean that the running time and the memory access pattern of the implementation are independent of secret material, therefore blocking attacks from timing side channels. The resulting signature scheme compares well with MQDSS and Picnic/Picnic2. Our scheme is much faster than MQDSS and Picnic/Picnic2 in terms of signing and verification, we have small public and private keys, and the signature sizes of our scheme are comparable to those of MQDSS and Picnic2. This makes our signature scheme based on PKP competitive with state of the art post-quantum signature schemes.

2 Preliminaries

2.1 The Permuted Kernel Problem (PKP)

The Permuted Kernel Problem (PKP) [23,10] is the problem on which the security of PKP-DSS is based. PKP is a linear algebra problem which asks to find a kernel vector of a given matrix under a vector-entries constraint. It's a generalization of the Partition problem [10, pg.224]. More precisely, it is defined as follows:

Definition 1 (Permuted Kernel Problem). *Given a finite field $\mathbb{F}_p$, a matrix $\mathbf{A} \in \mathbb{F}_p^{m \times n}$ and a n-vector $\mathbf{v} \in \mathbb{F}_p^n$, find a permutation $\pi \in S_n$ such that $\mathbf{A}\mathbf{v}_\pi = 0$, where $\mathbf{v}_\pi = (v_{\pi(1)}, \cdots, v_{\pi(n)})$*

A reduction of the 3-Partition problem proves PKP to be NP-Hard [10]. Moreover, solving random instances of PKP seems hard in practice. In fact, this is the fundamental design assumption of PKP-DSS. The hardness of PKP comes from, on the one hand, the big number of permutations, on the other hand, from the small number of possible permutations that satisfy the kernel equations. Note that, to make the problem more difficult, the n-vector $\mathbf{v}$ should have distinct coordinates. Otherwise if there are repeated entries, the space of permutations of $\mathbf{v}$ gets smaller. In the next section, we give the best known algorithm to solve the PKP problem.

Best known algorithms for solving PKP The implementation's efficiency of the first IDS, proposed by Shamir [23], based on PKP problem has led to several solving tools. There are various attacks for PKP, which are all exponential. We will not describe them here. Instead, we refer to [17] for further details. To estimate the concrete security of PKP, the authors of [17] review and compare the efficiency of the best known attacks in terms of the number of operations performed, for different finite fields. They bring together the Patarin-Chauvaud attack [21] and Poupard's algorithm [18] to provide an accurate program. The paper gives security estimates that we used to pick secure parameters sets for the Permuted Kernel Problem.

2.2 Commitment schemes

In our protocol, we use a commitment scheme $\mathsf{Com} : \{0,1\}^{\lambda} \times \{0,1\}^{\star} \to \{0,1\}^{2\lambda}$, that takes as input λ uniformly random bits bits, where λ is the security parameter, and a message $m \in \{0,1\}^{\star}$ and outputs a 2λ bit long commitment $\mathsf{Com}(\mathsf{bits}, m)$. In the description of our protocols, we often do not explicitly mention the commitment randomness. We write $\mathsf{S} \leftarrow \mathsf{Com}(m)$, to denote the process of picking a uniformly random bit string r, and setting $\mathsf{C} \leftarrow \mathsf{Com}(\mathsf{r}, m)$. Similarly, when we write check $\mathsf{C} = \mathsf{Com}(m)$, we actually mean that the prover communicates r to the verifier, and that the verifier checks if $\mathsf{C} = \mathsf{Com}(\mathsf{r}, m)$.

We assume that Com is computationally binding, which means that no computationally bounded adversary can produce a $\mathsf{r}, \mathsf{r}', m, m'$ with $m \neq m'$ such that $\mathsf{Com}(\mathsf{r}, m) = \mathsf{Com}(\mathsf{r}', m')$. We also assume that Com is computationally hiding, which means that for every pair of messages m, m', no computationally bounded adversary can distinguish the distributions of $\mathsf{Com}(m)$ and $\mathsf{Com}(m')$.

2.3 $2q$-Identification schemes and $2q$-extractors

In this paper, we will describe a so-called $2q$-Identification Scheme [24]. This is a 5-round identification scheme, where the first challenge is drawn uniformly at random from a challenge space of size q, and the second challenge is a random bit. Therefore, a transcript of an execution of a $2q$-protocol looks like $(\mathsf{com}, c, \mathsf{rsp}_1, b, \mathsf{rsp}_2)$. We now state the properties of a $2q$-protocol more formally:

Definition 2 ($2q$-Identification scheme,[24]). *A $2q$-Identification scheme is a canonical five-pass identification scheme* $(\mathsf{KeyGen}, \mathcal{P}, \mathcal{V})$ *with challenge spaces* $\mathcal{C}h_1$ *and* $\mathcal{C}h_2$ *for which it holds that* $|\mathcal{C}h_1| = q$ *and* $|\mathcal{C}h_2| = 2$. *Moreover, we require that the probability that the commitment* com *take a certain value is a negligible function of the security parameter.*

Definition 3 (Completeness). *An Identification scheme* $(\mathsf{KeyGen}, \mathcal{P}, \mathcal{V})$ *is called complete if when both parties follow the protocol honestly, the verifier accepts with probability 1. That is, we have*

$$\Pr\begin{bmatrix} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^{\lambda}) \\ \langle \mathcal{P}(\mathsf{sk}), \mathcal{V}(\mathsf{pk}) \rangle = 1 \end{bmatrix} = 1\,,$$

where $\langle \mathcal{P}(\mathsf{sk}), \mathcal{V}(\mathsf{pk})\rangle$ *stands for the common execution of the protocol between* $\mathcal{P}$ *with input* sk *and* $\mathcal{V}$ *with input* pk.

Definition 4 (Soundness with soundness error κ). *An identification scheme* $(\mathsf{KeyGen}, \mathcal{P}, \mathcal{V})$ *is called Sound with soundness error* κ*, if for any probabilistic polynomial time adversary* $\mathcal{A}$*, we have*

$$\Pr\begin{bmatrix}(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda) \\ \langle \mathcal{A}(1^\lambda, \mathsf{pk}), \mathcal{V}(\mathsf{pk})\rangle = 1\end{bmatrix} \leq \kappa + \epsilon(\lambda)\,,$$

for some negligible function $\epsilon(\lambda)$.

Definition 5 ((computational) Honest-Verifier Zero-Knowledge). *We say an identification scheme* $(\mathsf{KeyGen}, \mathcal{P}, \mathcal{V})$ *is HVZK if there exists a probabilistic polynomial time Simulator* $\mathcal{S}$ *that outputs transcripts that are computationally indistinguishable from transcripts of honest executions of the protocol.*

Finally, we define the notion of a $2q$-extractor, which is an algorithm that can extract the secret key from 4 transcripts that satisfy some properties. This is useful because when there exists a $2q$-extractor for a $2q$-Identification scheme, this implies that the identification scheme has soundness with knowledge error at most $\frac{q+1}{2q}$. Moreover, this implies that applying the Fiat-Shamir transform to the identification scheme results in a secure signature scheme.

Definition 6 ($2q$-extractability). *We say a* $2q$*-identification scheme* $(\mathsf{KeyGen}$, $\mathcal{P}$, $\mathcal{V})$ *has* $2q$*-extractability, if there exists a polynomial-time algorithm that given four transcripts* $(\mathsf{com}, c^{(i)}, \mathsf{rsp}_1^{(i)}, b^{(i)}, \mathsf{rsp}_2^{(i)})$ *for* i *from 1 to 4, such that*

$$\begin{gathered} c^{(1)} = c^{(2)} \neq c^{(3)} = c^{(4)} \\ \mathsf{rsp}_1^{(1)} = \mathsf{rsp}_1^{(2)} \quad \mathsf{rsp}_1^{(3)} = \mathsf{rsp}_1^{(4)} \\ b^{(1)} = b^{(3)} \neq b^{(2)} = b^{(4)} \end{gathered}$$

can efficiently extract a secret key.

Theorem 7 ([24], Theorem 3.1 and Theorem 4.3). *A* $2q$*-extractable* $2q$*-identification scheme is sound with knowledge error at most* $\frac{q+1}{2q}$*. Moreover, applying the Fiat-Shamir transform to such an identification scheme results in a EUF-CMA secure signature scheme in the Random Oracle Model.*

3 Identification scheme (IDS) based on PKP

In this section, we first present the 5-pass Zero-Knowledge Identification Scheme (ZK-IDS) based on the computational hardness of PKP [23,19], noted here PKP-IDS. Then, we introduce our optimized version of PKP-IDS and we prove that the optimized identification scheme is secure.

3.1 The original 5-pass PKP IDS

In this section, we present the original PKP-IDS [23,19], and we propose its slightly modified version. It consists of three probabilistic polynomial time algorithms $IDS = \big(\textsc{KeyGen},\ \mathcal{P},\ \mathcal{V}\big)$ which we will describe now.

Generation of the public key and secret key in PKP-IDS. The users first agree on a prime number p, and on n, m, the dimensions of the matrix $\mathbf{A}$. The public-key in PKP-IDS is an instance of PKP, a solution to this instance is the secret-key. Thus, the prover picks a (right) kernel-vector $\mathbf{w} \in \mathrm{Ker}(\mathrm{A})$, then randomly generates a secret permutation of n elements $\mathsf{sk} = \pi$ and finishes by computing $\mathbf{v} = \mathbf{w}_{\pi^{-1}}$. We summarize the key generation algorithm in Alg. 1.

Algorithm 1 KeyGen(n,m,p)

```
    $\mathbf{A} \xleftarrow{\$} \mathbb{F}_p^{m\times n}$
    $\mathbf{w} \xleftarrow{\$} \mathrm{Ker}(\mathbf{A})$
    $\pi \xleftarrow{\$} S_n$
    $\mathbf{v} \leftarrow \mathbf{w}_{\pi^{-1}}$
    **Return** $(\mathsf{pk} = (\mathbf{A}, \mathbf{v}), \mathsf{sk} = \pi)$
```

5-pass identification protocol: Prover $\mathcal{P}$ and Verifier $\mathcal{V}$.
The prover and verifier are interactive algorithms that realize the identification protocol in 5 passes. The 5 passes consist of one commitment and two responses transmitted from the prover to the verifier and two challenges transmitted from the verifier to the prover. The identification protocol is summarized in Alg. 2.

Theorem 8. *PKP-IDS is complete, moreover, if the used commitment scheme is computationally hiding then PKP-IDS is computationally honest-verifier zero-knowledge and if the commitment scheme is computationally binding, then PKP-IDS is sound with soundness error* $\kappa = \frac{p+1}{2p}$.

Algorithm 2 The original 5-pass PKP identification protocol

```
$\mathcal{P}(\mathsf{sk}, \mathsf{pk})$                                        $\mathcal{V}(\mathsf{pk})$

$\sigma \xleftarrow{\$} S_n$
$\mathbf{r} \xleftarrow{\$} \mathbb{F}_p^n$
$\mathsf{C}_0 \leftarrow \mathrm{Com}(\sigma, \mathbf{A}\mathbf{r})$
$\mathsf{C}_1 \leftarrow \mathrm{Com}(\pi\sigma, \mathbf{r}_\sigma)$
                              $\xrightarrow{\mathsf{C}_0,\mathsf{C}_1}$
                                              $c \xleftarrow{\$} \mathbb{F}_p$
                              $\xleftarrow{c}$
$\mathbf{z} \leftarrow \mathbf{r}_\sigma + c\mathbf{v}_{\pi\sigma}$
                              $\xrightarrow{\mathbf{z}}$
                                              $b \xleftarrow{\$} \{0,1\}$
                              $\xleftarrow{b}$
if $b = 0$ then
    $\mathsf{rsp} \leftarrow \sigma$
else
    $\mathsf{rsp} \leftarrow \pi\sigma$
end if
                              $\xrightarrow{\mathsf{rsp}}$
                                              if $b = 0$ then
                                                  accept if $\mathsf{C}_0 = \mathrm{Com}(\sigma, A\mathbf{z}_{\sigma^{-1}})$
                                              else
                                                  accept if $\mathsf{C}_1 = \mathrm{Com}(\pi\sigma, \mathbf{z} - c\mathbf{v}_{\pi\sigma})$
                                              end if
```

Proof. We refer to [23] for the complete proof.

In such ZK-IDS, it is usually possible to cheat if a cheater can correctly guess some challenges, so there is a nonzero probability (called the soundness error) that the verifier accepts the proof, even though the prover does not know the witness. In the case of PKP-IDS, this soundness error is $\frac{p+1}{2p}$. Thus, it is necessary to repeat the protocol several times to reduce the probability of fraud. Sequentially repeating the zero-knowledge proof N times results in an Identification scheme with knowledge error $\kappa_{\text{repeated}} = \kappa^N$, hence it suffices to repeat the protocol $\lceil \lambda / \log_2(\frac{2p}{p+1}) \rceil$ times to get a soundness error $\kappa \leq 2^{-\lambda}$. The systems are constructed such that executing the protocol does not reveal any secrets (Zero-knowledge).

3.2 The modified version of PKP-IDS

We now describe several optimizations to reduce the communication cost of the identification scheme, as well as the computational cost of the algorithms. We will start by explaining a few standard optimizations that are common for identification protocols based on zero-knowledge proofs. Then, we will explain some novel optimizations that apply to the specific context of PKP-IDS.

Hashing the commitments. In the commitment phase of the protocol, instead of transmitting all the $2N$ commitments $\mathsf{C}_0^{(1)}, \mathsf{C}_1^{(1)}, \cdots, \mathsf{C}_0^{(N)}, \mathsf{C}_1^{(N)}$ the prover can just hash all these commitments together with a collision resistant hash function $\mathcal{H}$ and only transmit the hash $h = \mathcal{H}(\mathsf{C}_0^{(1)}, \cdots, \mathsf{C}_1^{(N)})$. Then, the prover includes the N commitments $\mathsf{C}_{1-b_i}^{(i)}$ in the second response. Since the verifier can reconstruct the $\mathsf{C}_{b_i}^{i}$ himself, he now has all the $2N$ commitments, so he can hash them together and check if their hash matched h. With this optimization, we reduce the number of communicated commitments from $2N$ to N, at the cost of transmitting a single hash value.

Use seeds and PRG. Instead of directly choosing the permutation σ at random, we can instead choose a random seed of λ bits and use a PRG to expand this seed into a permutation σ. This way, instead of transmitting σ, we can just transmit the λ-bit seed. This reduces the communication cost per permutation from $\log_2(n!)$ bits to just λ bits. For example for 128-bits of security, we have $n = 69$, so the communication cost per permutation drops from $\log_2(69!) \approx 327$ bits to just 128 bits.

Matrix A in systematic form. Now we get to the PKP-IDS-specific optimizations. With high probability, we can perform elementary row operations on $\mathbf{A}$ to put it in the form $\big(I_m\ \mathbf{A}'\big)$, for some $(n-m)$-by-m matrix $\mathbf{A}'$. Since row operations do not affect the right kernel of A, we can just choose the matrix $\mathbf{A}$ of this form during key generation, without affecting the security of the scheme. This makes the protocol more efficient because multiplying by a matrix of this form requires only $(n-m)*m$ multiplications instead of $n*m$ multiplications for a general matrix multiplication.

Optimizing key generation. It is of course not very efficient to include in the public key the matrix $\mathbf{A} = \big[\mathbf{c_i^A},\ \mathbf{i} \in \{\mathbf{1}, \cdots, \mathbf{n}\}\big]$, where c_i^A is the i-th column of $\mathcal{A}$. The first idea is to just pick a random seed, and use a PRG to expand this seed to obtain the matrix $\mathbf{A}$. The public key then consists of a random seed, and the vector $\mathbf{v}$ of length n. However, we can do slightly better than this. We can use a seed to generate $\mathbf{A^*}$ which is formed by the first $n-1$ columns $c_1^A, \cdots, c_{n-1}^A$ of $\mathbf{A}$ and the vector v. Then we pick a random permutation π, and we solve for the last column c_n^A of $\mathbf{A}$ such that $\mathbf{v}_\pi$ is in the right kernel of $\mathbf{A}$. Now the public key only consists of a seed and a vector of length m (instead of a vector of length n). Another important advantage of this approach is that we do not need to do Gaussian elimination this way (and if fact this was the motivation behind this optimization). The optimized key generation procedure is given in Alg. 3.

Algorithm 3 KeyGen(n,m,p)

```
$sk.seed \leftarrow$ Randomly sample $\lambda$ $bits$
$(seed_\pi, pk.seed) \leftarrow \mathsf{PRG}_0(sk.seed)$
$\pi \leftarrow \mathsf{PRG}_1(seed_\pi)$
$(\mathbf{A}^*, \mathbf{v}) \leftarrow \mathsf{PRG}_2(pk.seed)$
Compute $\mathbf{c_n^A}$ from $A^*$ and $\mathbf{v}_\pi$
$\mathsf{sk} \leftarrow \mathbf{sk.seed}$
$\mathsf{pk} \leftarrow (\mathbf{pk.seed}, \mathbf{c_n^A})$
**Return** $(\mathsf{pk}, \mathsf{sk})$
```

Sending seeds instead of permutations. Because of the second optimization, we can send a λ-bit seed instead of σ, if the challenge bit $b=0$. However, in the case $b=1$, we still need to send the permutation $\pi\sigma$, because we cannot generate both σ and $\pi\sigma$ with a PRG. However, this problem can be solved. We can generate $\mathbf{r}_\sigma$ with a PRG, and then we can send this seed instead of $\pi\sigma$.

This seed can be used to compute $\pi\sigma$, because if the verifier knows $\mathbf{z}$ and $\mathbf{r}_\sigma$, then he can compute $\mathbf{z} - \mathbf{r}_\sigma = c\mathbf{v}_{\pi\sigma}$. And since $\mathbf{v}$ and c are known, it is easy to recover $\pi\sigma$ from $c\mathbf{v}_{\pi\sigma}$ (we choose the parameters such that the entries of v are all distinct, so there is a unique permutation that maps $\mathbf{v}$ to $\mathbf{v}_{\pi\sigma}$). Moreover, sending the seed for $\mathbf{r}_\sigma$ does not reveal more information than sending $\pi\sigma$, because given $\mathbf{z}$ and $\pi\sigma$ it is trivial to compute $\mathbf{r}_\sigma$, so this optimization does not affect the security of the scheme. However, there is a problem: If $c = 0$, then the $c\mathbf{v}_{\pi\sigma} = 0$, and so the verifier cannot recover $\pi\sigma$. To solve this problem we just restrict the challenge space to $\mathbb{F}_p \setminus \{0\}$. This increases the soundness error to $\frac{p}{2p-2}$ (instead of $\frac{p+1}{2p}$), but this is not a big problem. An important advantage of this optimization is that the signature size is now constant. Without this optimization, a response to the challenge $b = 0$ would be smaller than a response to $b = 1$. But with the optimization, the second response is always a random seed, regardless of the value of b. We summarize the one round of the optimized IDS modified version in Algorithm 4.

3.3 Security proof of the optimized Scheme

Theorem 9. – *The modified version of* PKP-IDS *is complete.*

– *If the commitment scheme is computationally binding, then the scheme is sound with soundness error $\kappa = \frac{p}{2p-2}$.*

– *If the used commitment scheme is computationally hiding and the output of* PRG_1 *and* PRG_2 *is indistinguishable from uniform randomness, then the scheme is computationally honest-verifier zero-knowledge.*

Proof. **Completeness.** In the case $b = 0$, if the prover acts honestly, then the commitment check will succeed if $\mathbf{Ar} = \mathbf{Az}_\sigma^{-1} = \mathbf{A}(\mathbf{r} + \mathbf{v}_{\pi\sigma\sigma^{-1}})$, which holds if and only if $\mathbf{Av}_\pi = 0$. Therefore, if π is a solution to the PKP problem, then the verifier will accept the transcript. In an honest execution with $b = 1$ the verifier will always accept, regardless of whether π was a solution to the PKP problem or not.

Soundness. First, we prove that the scheme has a $q2$-extractor [24]. That is, we show that, given four accepted transcripts $(\mathsf{C}_0, \mathsf{C}_1, c^{(i)}, \mathbf{z}^{(i)}, b^{(i)}, \mathsf{rsp}^{(i)})$ for i from 1 to 4, such that

$$c^{(1)} = c^{(2)} \neq c^{(3)} = c^{(4)}$$

$$\mathbf{z}^{(1)} = \mathbf{z}^{(2)} \quad \mathbf{z}^{(3)} = \mathbf{z}^{(4)}$$

$$b^{(1)} = b^{(3)} \neq b^{(2)} = b^{(4)}$$

Algorithm 4 The modified 5-pass of PKP-IDS

```
$\mathcal{P}(\mathsf{sk}, \mathsf{pk})$                                   $\mathcal{V}(\mathsf{pk})$

$\mathsf{seed}_0, \mathsf{seed}_1 \xleftarrow{\$} \{0,1\}^\lambda$
$\sigma \leftarrow \mathsf{PRG}_1(seed_\sigma)$
$\mathbf{r}_\sigma \leftarrow \mathsf{PRG}_2(\mathbf{r}_\sigma.seed)$
$\mathsf{C}_0 \leftarrow \mathrm{Com}(\sigma, \mathbf{Ar})$
$\mathsf{C}_1 \leftarrow \mathrm{Com}(\pi\sigma, \mathbf{r}_\sigma)$
                    $\xrightarrow{\mathsf{C}_0, \mathsf{C}_1}$
                                         $c \xleftarrow{\$} \mathbb{F}_p \setminus \{0\}$
                    $\xleftarrow{c}$
$\mathbf{z} \leftarrow \mathbf{r}_\sigma + c\mathbf{v}_{\pi\sigma}$
                    $\xrightarrow{\mathbf{z}}$
                                         $b \xleftarrow{\$} \{0,1\}$
                    $\xleftarrow{b}$
$\mathsf{rsp} \leftarrow \mathsf{seed}_b$
                    $\xrightarrow{\mathsf{rsp}}$
                                         if $b = 0$ then
                                             $\sigma \leftarrow \mathsf{PRG}_1(\mathsf{rsp})$
                                             accept if $\mathsf{C}_0 = \mathrm{Com}(\sigma, A\mathbf{z}_{\sigma^{-1}})$
                                         else
                                             $\mathbf{r}_\sigma \leftarrow \mathsf{PRG}_2(\mathsf{rsp})$
                                             if $\mathbf{z} - \mathbf{r}_\sigma$ is not a permutation of $c\mathbf{v}$ then
                                                 Return reject
                                             else
                                                 Let $\rho \in S_n$ such that $c\mathbf{v}_\rho = \mathbf{z} - \mathbf{r}_\sigma$.
                                             end if
                                             accept if $\mathsf{C}_1 = \mathrm{Com}(\rho, \mathbf{r}_\sigma)$
                                         end if
```

one can efficiently extract a solution for the PKP problem.

By relabeling the transcripts if necessary, we can assume that $b^{(1)} = b^{(3)} = 0$ and $b^{(2)} = b^{(4)} = 1$. Let us first look at transcripts 1 and 3. Let $\sigma = \mathsf{PRG}_1(\mathsf{rsp}^{(1)})$ and $\sigma' = \mathsf{PRG}_1(\mathsf{rsp}^{(3)})$, and let $\mathbf{x} = \mathbf{A}\mathbf{z}^{(1)}_{\sigma^{-1}}$ and $\mathbf{x}' = \mathbf{A}\mathbf{z}^{(3)}_{\sigma'^{-1}}$. Then, because both transcripts are accepted, we have

$$\mathsf{C}_0 = \mathsf{Com}(\sigma, \mathbf{x}) = \mathsf{Com}(\sigma', \mathbf{x}').$$

Therefore, the computationally binding property of Com implies that with overwhelming probability we have $\sigma = \sigma'$ and $\mathbf{x} = \mathbf{x}'$.

Now, lets look at transcripts 2 and 4. Let $\mathbf{y} = \mathsf{PRG}_2(\mathsf{rsp}^{(2)})$ and $\mathbf{y}' = \mathsf{PRG}_2(\mathsf{rsp}^{(4)})$. Since both transcripts are accepted, we know that $\mathbf{z}^{(2)} - \mathbf{y}$ and $\mathbf{z}^{(4)} - \mathbf{y}'$ are permutations of $c^{(2)}\mathbf{v}$ and $c^{(4)}\mathbf{v}$ respectively. Let ρ and ρ' be the permutations such that $c^{(2)}\mathbf{v}_\rho = \mathbf{z}^{(2)} - \mathbf{y}$ and $c^{(4)}\mathbf{v}'_\rho = \mathbf{z}^{(4)} - \mathbf{y}'$. Since both transcripts are accepted, we have

$$\mathsf{C}_1 = \mathsf{Com}(\rho, y) = \mathsf{Com}(\rho', y')\,,$$

so the computationally binding property of Com implies that with overwhelming probability we have $\rho = \rho'$ and $\mathbf{y} = \mathbf{y}'$. Now, we put everything together to get

$$\begin{aligned} 0 &= \mathbf{A}(\mathbf{z}^{(1)}_{\sigma^{-1}} - \mathbf{z}^{(3)}_{\sigma^{-1}}) \\ &= \mathbf{A}(\mathbf{z}^{(2)}_{\sigma^{-1}} - \mathbf{z}^{(4)}_{\sigma^{-1}}) \\ &= \mathbf{A}(c^{(2)}\mathbf{v}_{\rho\sigma^{-1}} - \mathbf{y}_{\sigma^{-1}} - c^{(4)}\mathbf{v}_{\rho\sigma^{-1}} + \mathbf{y}_{\sigma^{-1}}) \\ &= (c^{(2)} - c^{(4)})\mathbf{A}\mathbf{v}_{\rho\sigma^{-1}}\,. \end{aligned}$$

Since $c^{(2)} - c^{(4)}$ is nonzero, this means that $\rho\sigma^{-1}$ is a solution to the permuted kernel problem. Moreover the extractor can efficiently extract this solution, because he can extract ρ from either transcript 2 or 4, and he can extract σ from either transcript 1 or 3.

It is known that $2q$-extractability implies soundness with error $\frac{q+1}{2q}$, where q is the size of the first challenge space [22,24]. In our case, the first challenge space has $p-1$ elements, so the optimized IDS has soundness error $\frac{p}{2p-2}$.

Honest-Verifier Zero-knowledge. To prove Honest-Verifier Zero-Knowledge we construct a simulator that outputs transcripts that are computationally indistinguishable from transcripts of honest executions of the identification scheme. First, the simulator picks a uniformly random value $c \in \mathbb{F}_p \setminus \{0\}$ and a uniformly random bit b. We treat the cases $b = 0$ and $b = 1$ separately.

Case $b = 0$ **:** The simulator picks a random seed seed_0, a uniformly random vector $\mathbf{z}$, and computes $\sigma = \mathsf{PRG}_1(\mathsf{seed}_0)$ and $\mathsf{C}_0 = \mathsf{Com}(\sigma, \mathbf{Az})$. The simulator also commits to a dummy value to get C_1. Now the simulator outputs $(\mathsf{C}_0, \mathsf{C}_1, c, \mathbf{z}, b, \mathsf{seed}_\sigma)$.

This distribution is indistinguishable from honestly generated transcripts with $b = 0$. Indeed, the values $c, \mathbf{z}, \mathsf{seed}_0$ are indistinguishable from uniformly random in both the simulated transcripts and the honest transcripts (here we use the assumption that the output of PRG_2 is indistinguishable from the uniform distribution). The first commitment $\mathsf{C}_0 = \mathsf{Com}(\sigma, \mathbf{Az}_{\sigma^{-1}})$ is a function of seed_0 and $\mathbf{z}$, so it also has the same distribution in the simulated and the honest transcripts. Finally, the commitment C_1 is never opened, so the computationally hiding property of Com guarantees that C_1 in the simulated transcript is computationally indistinguishable from the C_1 in an honest transcript.

Case $b = 1$ **:** The simulator picks a uniformly random seed seed_1 and a uniformly random permutation ρ and computes $\mathbf{r}_\sigma = \mathsf{PRG}_2(\mathsf{seed}_1)$, $z = c\mathbf{v}_\rho + \mathbf{r}_\sigma$ and $\mathsf{C}_1 = \mathsf{Com}()$. The simulator also commits to a dummy value to produce a commitment C_0, then the simulator outputs the transcript $(\mathsf{C}_0, \mathsf{C}_1, c, \mathbf{z}, b, \mathsf{seed}_1)$.

We now show that the simulated transcripts are indistinguishable from honestly generated transcripts with $b = 1$. It is clear that c and seed_1 are uniformly random in both the simulated transcripts and the honestly generated transcripts. Moreover, in both the simulated and the real transcripts, $\mathbf{z}$ is equal to $\mathsf{PRG}_2(\mathsf{seed}_1) + c\mathbf{v}_\rho$, and $\mathsf{C}_1 = \mathsf{Com}(\rho, \mathsf{PRG}_2(\mathsf{seed}_1))$ where ρ is indistinguishable from a uniformly random permutation (here we need the assumption that the output of PRG_1 is indistinguishable from a uniformly random permutation). Therefore $\mathbf{z}$ and C_1 have the same distribution in the simulated and the honest transcripts. Finally, the computationally hiding properties of Com guarantee that the value of C_0 in the simulated transcripts is indistinguishable from that of C_0 in honestly generated transcripts.

3.4 Communication cost

We can now provide the communication complexity of N rounds of the modified IDS, of which the soundness error is $\left(\frac{p}{2p-2}\right)^N$. The commitment consists of a single hash value, which is only 2λ bits. The first response consists of N vectors of length n over $\mathbb{F}_p$, so this costs $Nn\lceil \log_2 p \rceil$ bits of communication. Lastly, the second responses consist of N random λ-bit seeds, N commitments (which consist of 2λ bits each) and N commitment random strings (which consist of λ

bits each), so this costs $4N\lambda$ bits of communication. In total, the communication cost (ignoring the challenges) is

$$2\lambda + N\left(n\lceil\log_2 p\rceil + 4\lambda\right) \ .$$

4 Digital signature scheme (DSS) based on PKP

We present here the main contribution of this work which is to construct a digital signature scheme, based on the PKP problem, from the optimized IDS defined in Section 3. This is simply a direct application of the well-known Fiat Shamir transformation [9].

The key generation algorithm is identical to the key generation algorithm for the identification scheme. To sign a message m, the signer executes the first phase of the commitment scheme to get a commitment com. Then he derives the first challenge $\mathbf{c} = (c_1, \cdots, c_N)$ from m and com by evaluating a hash function $\mathcal{H}_1(m||\mathsf{com})$. Then he does the next phase of the identification protocol to get the N response vectors $\mathsf{rsp_1} = (\mathbf{z}^{(1)}, \cdots, \mathbf{z}^{(N)})$. Then he uses a second hash function to derive $\mathbf{b} = (b_1, \ldots, b_N)$ from m, com and $\mathsf{rsp_1}$ as $\mathcal{H}_2(m||\mathsf{com}, \mathsf{rsp_1})$. Then he finishes the identification protocol to obtain the vector of second responses $\mathsf{rsp}_2 = (\mathsf{rsp}^{(1)}, \cdots, \mathsf{rsp}^{(N)})$. Then, the signature is simply $(\mathsf{com}, \mathsf{rsp}_1, \mathsf{rsp}_2)$.

To verify a signature $(\mathsf{com}, \mathsf{rsp}_1, \mathsf{rsp}_2)$ for a message m, the verifier simply uses the hash function $\mathcal{H}_1$ and $\mathcal{H}_2$ to obtain $\mathbf{c}$ and $\mathbf{b}$ respectively. Then, he verifies that $(\mathsf{com}, \mathbf{c}, \mathsf{rsp}_1, \mathbf{b}, \mathsf{rsp}_2)$ is a valid transcript of the identification protocol.

The signing and verification algorithms are displayed in Algorithm 5 and 6 in more detail.

We then get the same security result as Th. 5.1 in [7], with the same proof.

Theorem 10. *PKP-DSS is Existential-Unforgeable under Chosen Adaptive Message Attacks (EU-CMA) in the random oracle model, if*

- *the search version of the Permuted Kernel problem is intractable,*
- *the hash functions and pseudo-random generators are modeled as random oracles,*
- *the commitment functions are computationally binding, computationally hiding, and the probability that their output takes a given value is negligible in the security parameter.*

Algorithm 5 Sign(sk, m)

```
1: derive $\mathbf{A}, \mathbf{v}$ and $\pi$ from $\mathsf{sk}$.
2: for $i$ from 1 to $N$ do
3:     pick $\lambda$-bit seeds $\mathsf{seed}_0^{(i)}$ and $\mathsf{seed}_1^{(i)}$ uniformly at random
4:     $\sigma^{(i)} \leftarrow \mathsf{PRG}_1(\mathsf{seed}_0^{(i)})$
5:     $\mathbf{r}_\sigma^{(i)} \leftarrow \mathsf{PRG}_2(\mathsf{seed}_1^{(i)})$
6:     $\mathsf{C}_0^{(i)} = \mathrm{Com}\big(\sigma^{(i)}, \mathbf{A}\mathbf{r}^{(i)}\big)$,
7:     $\mathsf{C}_1^{(i)} = \mathrm{Com}\big(\pi\sigma^{(i)}, \mathbf{r}_\sigma^{(i)}\big)$.
8: end for
9: $\mathsf{com} := \mathcal{H}_{\mathsf{com}}\big(\mathsf{C}_0^{(1)},\ \mathsf{C}_1^{(1)}, \cdots, \mathsf{C}_0^{(N)},\ \mathsf{C}_1^{(N)}\big)$
10: $c^{(1)}, \cdots, c^{(N)} \leftarrow \mathcal{H}_1\big(m||\mathsf{com}\big)$.            $c^i \in \mathbb{F}_p \setminus \{0\}$
11: for $i$ from 1 to $N$ do
12:     $\mathbf{z}^{(i)} \leftarrow \mathbf{r}_\sigma^{(i)} + c^{(i)}\mathbf{v}_{\pi\sigma^{(i)}}$
13: end for
14: $\mathsf{rsp}_1 \leftarrow (\mathbf{z}^{(1)}, \cdots, \mathbf{z}^{(N)})$
15: $b^{(1)}, \cdots, b^{(N)} \leftarrow \mathcal{H}_2(m||\mathsf{com}||\mathsf{rsp}_1)$
16: for $i$ from 1 to $N$ do
17:     $\mathsf{rsp}_2^{(\mathsf{i})} \leftarrow (\mathsf{seed}_{b^{(i)}}^{(i)}||\mathsf{C}_{1-b^{(i)}}^{(i)})$
18: end for
19: $\mathsf{rsp}_2 \leftarrow (\mathsf{rsp}_2{}^{(1)}, \cdots, \mathsf{rsp}_2{}^{(N)})$
20: Return $(\mathsf{com}, \mathsf{rsp}_1, \mathsf{rsp}_2)$
```

Algorithm 6 Verify($m, \mathsf{pk}, \sigma = (\mathsf{com}, \mathsf{rsp}_1, \mathsf{rsp}_2)$)

```
1: $c^{(1)}, \cdots, c^{(N)} \leftarrow \mathcal{H}_1\big(m||\mathsf{com}\big)$.
2: $b^{(1)}, \cdots, b^{(N)} \leftarrow \mathcal{H}_2(m||\mathsf{com}||\mathsf{rsp}_1)$
3: Parse $\mathsf{rsp}_1$ as $\mathbf{z}^{(1)}, \cdots, \mathbf{z}^{(N)}$
4: Parse $\mathsf{rsp}_2$ as $\mathsf{seed}^{(1)}, \cdots, \mathsf{seed}^{(N)}, \mathsf{C}_{1-b^{(1)}}^{(1)}, \cdots, \mathsf{C}_{1-b^{(N)}}^{(N)}$
5: for $i$ from 1 to $N$ do
6:     if $b^{(i)} = 0$ then
7:         $\sigma^{(i)} \leftarrow \mathsf{PRG}_1(\mathsf{seed}^{(i)})$
8:         $\mathsf{C}_0^{(i)} \leftarrow \mathrm{Com}(\sigma^{(i)}, \mathbf{A}\mathbf{z}_{\sigma^{(i)-1}})$
9:     else
10:        $\mathbf{r}_\sigma^{(i)} \leftarrow \mathsf{PRG}_2(\mathsf{seed}^{(i)})$
11:        if $\mathbf{z}^{(i)} - \mathbf{r}_\sigma$ is not a permutation of $c\mathbf{v}$ then
12:            Return reject
13:        else
14:            $\pi\sigma^{(i)} \leftarrow$ the permutation that maps $c\mathbf{v}$ to $\mathbf{z}^{(i)} - \mathbf{r}_\sigma$.
15:        end if
16:        $\mathsf{C}_1^{(i)} \leftarrow \mathrm{Com}(\pi\sigma^{(i)}, \mathbf{r}_\sigma^{(i)})$
17:    end if
18: end for
19: $\mathsf{com}' := \mathcal{H}_{\mathsf{com}}\big(\mathsf{C}_0^{(1)},\ \mathsf{C}_1^{(1)}, \cdots, \mathsf{C}_0^{(N)},\ \mathsf{C}_1^{(N)}\big)$
20: Return accept if and only if $\mathsf{com} = \mathsf{com}'$
```

4.1 Generic attack

If the number of iterations N is chosen such that $(\frac{p}{2p-2})^N \leq 2^{-\lambda}$, then the cheating probability of the identification protocol is bounded by $2^{-\lambda}$. However, a recent attack by Kales and Zaverucha on MQDSS reveals that this does not meant that the Fiat-Shamir signature scheme has λ bits of security [16]. They give a generic attack that also applies to PKP-DSS. The attack exploits the fact that if an attacker can guess the first challenge **or** the second challenge, he can produce responses that the verifier will accept. The idea is to split up the attack in two phases. In the first phase, the attacker guesses the values of the N first challenges, and uses this guess to produce commitments. Then, he derives the challenges from the commitment and he hopes that at least k of his N guesses are correct. This requires on average

$$\mathsf{Cost}_1(N,k) = \sum_{i=k}^{N} \left(\frac{1}{p-1}\right)^k \left(\frac{p-2}{p-1}\right)^{N-k} \binom{N}{k}$$

trials. In the second phase, the attacker guesses the values of second challenges, and uses these guesses to generate a response. Then he derives the second challenges with a hash function and he hopes that his guess was correct for the $N-k$ rounds of the identification protocol where he did not guess the first challenge correctly. This requires on average 2^{N-k} tries. Therefore, the total cost of the attack is

$$\min_{0\leq k\leq N} \mathsf{Cost}_1(N,k) + 2^{N-k}\,.$$

5 Parameter choice and Implementation

5.1 Parameter choice

The PKP-DSS is mainly affected by the following set of parameters: (p, n, m). We now explicitly detail the choice of these parameters. Recall that firstly the IDS [23] was designed to suit small devices. Thus, Shamir proposed $p = 251$. To have an efficient implementation we choose p to be a prime number close to a power of 2, such as 251, 509 and 4093. A solution of a random instance of PKP is to find a kernel n-vector $(\mathbf{v}_\pi)$ with distinct coordinates in $\mathbb{F}_p$. Hence, the probability to find such a vector shouldn't be too small. The probability of an arbitrary vector to be in the kernel of the matrix $A \in \mathcal{M}_{m\times n}$ whose rank is equal to m, is p^{-m}. Moreover, if the n-vector v has no repeated entries, its orbit under the possible permutations π contains $n!$ vectors. Thus, to get on average one solution, we have the following constraint: $n! \approx p^m$. And finally, using the complexity of Poupard's algorithm [18] combined with Patarin-Chauvaud's method (See Section 2.1), triplets of (p, n, m) were selected matching the security requirements and optimizing the size of the signature. With these parameter choices, the scheme is secure against all the attacks described in [17]. We pick the value of N just large enough such that

$$\min_{0 \leq k \leq N} \mathsf{Cost}_1(N, k) + 2^{N-k} \geq 2^\lambda ,$$

such that the scheme is secure against the generic attack of Kales and Zaverucha [16]. The chosen parameter sets for three different security levels are shown in Table 1.

Parameter Set	Security level	p	n	m	Iterations N	Attack cost
PKP-DSS-128	128	251	69	41	157	2^{130}
PKP-DSS-192	192	509	94	54	229	2^{193}
PKP-DSS-256	256	4093	106	47	289	2^{257}

Table 1. PKP-DSS Parameters sets

5.2 Key and signature sizes

Public key. A public key consists of the last column $\mathbf{c_n^A}$ of $\mathbf{A}$ and a random seed **pk.seed**, which is used to generate $\mathbf{A^*}$ which is formed by all but the last column of $\mathbf{A}$ and the vector $\mathbf{v}$. Therefore, the public key consist of $\lambda + m\lfloor \log_2(p) \rfloor$ bits.

Secret key. A secret key is just a random seed **pk.seed** that was used to seed the key generation algorithm, therefore it consists of only λ bits.

Signature. Finally, a signature consists of a transcript of the identification protocol (excluding the challenges, because they are computed with a hash function). In Sect 3.4 we calculated that a transcript can be represented with $2\lambda + N\left(n\lceil \log_2 p \rceil + 4\lambda\right)$ bits, so this is also the signature size.

In Table 2 we summarize the key and signature sizes for the parameter sets proposed in the previous section.

Security level	Parameters (p, n, m, N)	$\vert\mathsf{sk}\vert$ Bytes	$\vert\mathsf{pk}\vert$ Bytes	$\vert\mathsf{sig}\vert$ Kilobytes
128	$(251, 69, 41, 157)$	16	57	20.4
192	$(509, 94, 54, 229)$	24	85	45.2
256	$(4093, 106, 47, 289)$	32	103	81.1

Table 2. Key and signature sizes for PKP-DSS with the three proposed parameter sets.

5.3 Implementation

To showcase the efficiency of PKP-DSS and to compare the performance to existing Fiat-Shamir signatures we made a proof-of-concept implementation in plain C. The code of our implementation is available on GitHub at [4]. We have used SHA-3 as hash function and commitment scheme, and we have used SHAKE128 as extendable output function. The running time of the signing and verification algorithms is dominated by expanding seeds into random vectors and random permutations. This can be sped up by using a vectorized implementation of SHAKE128, and using vector instructions to convert the random bitstring into a vector over $\mathbb{F}_p$ or a permutation in S_n. We leave this task for future work.

Making the implementation constant time. Most of the key generation and signing algorithms is inherently constant time (signing branches on the value of the challenge bits b, but this does not leak information because b is public). The only problem was that applying a secret permutation to the entries of a vector, when implemented naively, involves accessing data at secret indices. To prevent this potential timing leak we used the "djbsort" constant time sorting code [3]. More specifically, (see Alg. 7) we combine the permutation and the vector into a single list of n integers, where the permutation is stored in the most significant bits, and the entries of the vector are stored in the least significant bits. Then we sort this list of integers in constant time and we extract the permuted vector from the low order bits. Relative to the naive implementation this slows down signing by only 11%. There is no significant slowdown for key generation.

5.4 Performance results

To measure the performance of our implementation we ran experiments on a laptop with a i5-8250U CPU running at 1.8 GHz. The C code was compiled

Algorithm 7 Constant time computation of $v' = v_\sigma$

```
1: Initialize a list of integers $L \leftarrow \emptyset$
2: $L := \big[\sigma[1] * B + v[1], \cdots, \sigma[n] * B + v[n]\big]$, where $B > n$ is a constant
3: sort $L$ in constant time
4: $v' := \big[L[1]\ mod\ B, \cdots, L[n]\ mod\ B\big]$
5: Return $v'$
```

with gcc version 7.4.0 with the compile option -O3. The cycle counts in Table 3 are averages of 10000 key generations, signings, and verifications.

Security level	Parameters (p, n, m, N)	KeyGen 10^3 cycles	Sign 10^3 cycles	Verify 10^3 cycles
128	$(251, 69, 41, 157)$	72	2518	896
192	$(509, 94, 54, 229)$	121	5486	2088
256	$(4093, 106, 47, 289)$	151	7411	3491

Table 3. Average cycle counts for key generation, signing and verification, for our implementation of PKP-DSS with the three proposed parameter sets.

5.5 Comparison with existing FS signatures

In Table 4, we compare PKP-DSS to MQDSS, Picnic, and Picnic2. We can see that for all the schemes the public and secret keys are all very small. The main differences are signature size and speed. When compared to MQDSS, the signature sizes of PKP-DSS are roughly 30% smaller, while being a factor 14 and 30 faster for signing and verification respectively. Compared to Picnic, the PKP-DSS signatures are roughly 40% smaller, and signing and verification are 4 and 9 times faster respectively. Compared to Picinc2 our scheme is 153 and 170 times faster for signing and verification, but this comes at the cost of 50% larger signatures. Finally, compared to SUSHSYFISH [12], a different scheme based on the Permuted Kernel Problem, our scheme is 3.4 and 6.6 times faster, but at the cost of 45% larger signatures.

Security level	Scheme	Secret key (Bytes)	Public key (Bytes)	Signature (KBytes)	Sign 10^6 cycles	Verify 10^6 cycles
128	PKP-DSS-128	16	57	20.4	2.5	0.9
	MQDSS-31-48	16	46	28.0	36	27
	Picnic-L1-FS	16	32	33.2	10	8.4
	Picnic2-L1-FS	16	32	13.5	384	153
	SUSHSYFISH-1	16	72	14.0	8.6	6
192	PKP-DSS-192	24	85	45.2	5.5	2.1
	MQDSS-31-64	24	64	58.6	116	85
	Picnic-L3-FS	24	48	74.9	24	20
	Picnic2-L3-FS	24	48	29.1	1183	357
	SUSHSYFISH-3	24	108	30.8	22.7	16.5
256	PKP-DSS-256	32	103	81.1	7.4	3.5
	Picnic-L5-FS	32	64	129.7	44	38
	Picnic2-L5-FS	32	64	53.5	2551	643
	SUSHSYFISH-5	32	142	54.9	25.7	18

Table 4. Comparison of different post-quantum Fiat-Shamir schemes

6 Conclusion

We introduced a new post-quantum secure signature scheme PKP-DSS, which is based on a PKP Zero-knowledge identification scheme [23]. We optimized this identification scheme, and to make it non-interactive, we used the well-known Fiat-Shamir transform.

We developed a constant-time implementation of PKP-DSS and we conclude that our scheme is competitive with other Post-Quantum Fiat-Shamir signature schemes such as MQDSS, Picnic/Picnic2, and SUSHSYFISH. The main advantages of our scheme are that signing and verification are much faster than existing Fiat-Shamir signatures and that the scheme is very simple to implement. Our implementation takes only 440 lines of C code.

Acknowledgments. This work was supported by the European Commission through the Horizon 2020 research and innovation program under grant agreement H2020-DS-LEIT-2017-780108 FENTEC, by the Flemish Government through FWO SBO project SNIPPET S007619N and by the IF/C1 on Cryptanalysis of post-quantum cryptography and by the French Programme d'Investissement d'Avenir under national project RISQ P141580. Ward Beullens is funded by an FWO fellowship.

References

1. Baritaud, T., Campana, M., Chauvaud, P., Gilbert, H. On the security of the permuted kernel identification scheme. In Annual International Cryptology Conference (pp. 305-311). (1992, August), Springer, Berlin, Heidelberg.
2. Bennett, C. H., Bernstein, E., Brassard, G., Vazirani, U. (1997). Strengths and weaknesses of quantum computing. SIAM journal on Computing, 26(5), 1510-1523.
3. The djbsort software library for sorting arrays of integers or floating-point numbers in constant time. `https://sorting.cr.yp.to/`
4. Beullens, Ward. PKPDSS, (2019) Public GitHub repository. `https://github.com/WardBeullens/PKPDSS`
5. Beullens, Ward. On sigma protocols with helper for MQ and PKP, fishy signature schemes and more. Cryptology ePrint Archive, Report 2019/490, 2019. `https://eprint.iacr.org/2019/490`.
6. Chase, M., Derler, D., Goldfeder, S., Orlandi, C., Ramacher, S., Rechberger, C., ... Zaverucha, G. Post-quantum zero-knowledge and signatures from symmetric-key primitives. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (pp. 1825-1842) (2017, October). ACM.
7. Chen, M. S., Hülsing, A., Rijneveld, J., Samardjiska, S., Schwabe, P. MQDSS specifications. (2018).
8. Damgård, I. Commitment schemes and zero-knowledge protocols. In School organized by the European Educational Forum (pp. 63-86). (1998, June). Springer, Berlin, Heidelberg.
9. Fiat, A., Shamir, A. (1986, August). How to prove yourself: Practical solutions to identification and signature problems. In Advances in Cryptology—CRYPTO'86 (pp. 186-194). Springer, Berlin, Heidelberg.
10. Gary, M., Johnson, D. (1979). Computers and Intractability: A Guide to NP-Completeness. New York: W H.
11. Georgiades, J. (1992). Some remarks on the security of the identification scheme based on permuted kernels. Journal of Cryptology, 5(2), 133-137.
12. W. Beullens. On sigma protocols with helper for MQ and PKP, fishy signature schemes and more IACR Cryptology ePrint Archive 2019 (2019): `https://eprint.iacr.org/2019/490`.
13. Don, J., Fehr, S., Majenz, C., Schaffner, C. Security of the Fiat-Shamir transformation in the Quantum Random-Oracle Model. Cryptology ePrint Archive, Report 2019/190 (2019), `https://eprint.iacr.org/2019/190`
14. Haitner, I., Nguyen, M. H., Ong, S. J., Reingold, O., Vadhan, S. Statistically hiding commitments and statistical zero-knowledge arguments from any one-way function. SIAM Journal on Computing, 39(3), pp. 1153-1218.
15. Jaulmes, É., Joux, A. Cryptanalysis of PKP: a new approach. In International Workshop on Public Key Cryptography (pp. 165-172). (2001, February) Springer, Berlin, Heidelberg.
16. Daniel Kales, Greg Zaverucha. "Forgery attacks against MQDSSv2.0" Note postes on the NIST PQC forum `https://groups.google.com/a/list.nist.gov/forum/?utm_medium=email&utm_source=footer#!msg/pqc-forum/LlHhfwg73eQ/omM6TWwlEwAJ`

17. Koussa, Eliane, Gilles Macario-Rat, and Jacques Patarin. "On the complexity of the Permuted Kernel Problem." IACR Cryptology ePrint Archive 2019 (2019): 412.
18. Poupard, G., (1997). A realistic security analysis of identification schemes based on combinatorial problems. European Transactions on Telecommunications.
19. Lampe, R., Patarin, J. Analysis of Some Natural Variants of the PKP Algorithm. IACR Cryptology ePrint Archive, 2011, 686.
20. NIST categories: Security strength categories. `https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf`
21. Patarin, J., Chauvaud, P. Improved algorithms for the permuted kernel problem. In Annual International Cryptology Conference (pp. 391-402). (1993, August) Springer, Berlin, Heidelberg.
22. Sakumoto, K., Shirai, T., Hiwatari, H. (2011, August). Public-key identification schemes based on multivariate quadratic polynomials. In Annual Cryptology Conference (pp. 706-723). Springer, Berlin, Heidelberg.
23. Shamir, A. (1989, August). An efficient identification scheme based on permuted kernels. In Conference on the Theory and Application of Cryptology (pp. 606-609). Springer, New York, NY.
24. Chen, Ming-Shing, et al. From 5-Pass $\mathcal{MQ}$-Based Identification to $\mathcal{MQ}$-Based Signatures. International Conference on the Theory and Application of Cryptology and Information Security. Springer, Berlin, Heidelberg, 2016.
25. Kiltz, E., Lyubashevsky, V., Schaffner, C. A new identification scheme based on syndrome decoding. In Annual International Conference on the Theory and Applications of Cryptographic Techniques (pp. 552-586). (2018) Springer, Cham.
26. Unruh, D. Post-quantum security of Fiat-Shamir. In International Conference on the Theory and Application of Cryptology and Information Security. pp. 65-95. (2017, December) Springer, Cham.
27. Unruh, D. Non-interactive zero-knowledge proofs in the quantum random oracle model. In Annual International Conference on the Theory and Applications of Cryptographic Techniques (pp. 755-784). (2015, April). Springer, Berlin, Heidelberg.
28. Gaëtan Leurent and Phong Q. Nguyen, How Risky is the Random-Oracle Model?
29. Shai Halevi and Hugo Krawczyk, Strengthening Digital Signatures Via Randomized Hashing, In CRYPTO,2006,41–59

Chapter 7

Cryptanalysis of the Legendre PRF

Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin.

– John Von Neumann

Publication Data

Beullens, W., Beyne, T., Udovenko, A., and Vitto, G. Cryptanalysis of the Legendre PRF and Generalizations. IACR Transactions on Symmetric Cryptology, 2020(1), pages 313-330, 2020.

My Contribution

Tim Beyne and I discovered the $\tilde{O}(p/M^2)$ attack that is central to the paper and solved the first ETH challenge. Aleksei Udovenko and Guiseppe Vitto independently discovered the same attack, and beat us to the second challenge. We decided to join forces and write the paper and improve the results together. All authors contributed equally.

Cryptanalysis of the Legendre PRF and Generalizations*

Ward Beullens[1], Tim Beyne[1],
Aleksei Udovenko[2] and Giuseppe Vitto[2]

[1] imec-COSIC, ESAT, KU Leuven, Belgium

[2] SnT, University of Luxembourg, Luxembourg

Abstract. The *Legendre PRF* relies on the conjectured pseudorandomness properties of the Legendre symbol with a hidden shift. Originally proposed as a PRG by Damgård at CRYPTO 1988, it was recently suggested as an efficient PRF for multiparty computation purposes by Grassi *et al.* at CCS 2016. Moreover, the Legendre PRF is being considered for usage in the Ethereum 2.0 blockchain.

This paper improves previous attacks on the Legendre PRF and its higher-degree variant due to Khovratovich by reducing the time complexity from $\mathcal{O}(p \log p/M)$ to $\mathcal{O}(p \log^2 p/M^2)$ Legendre symbol evaluations when $M \leq \sqrt[4]{p \log^2 p}$ queries are available. The practical relevance of our improved attack is demonstrated by breaking three concrete instances of the PRF proposed by the Ethereum foundation. Furthermore, we generalize our attack in a nontrivial way to the higher-degree variant of the Legendre PRF and we point out a large class of weak keys for this construction.

Lastly, we provide the first security analysis of two additional generalizations of the Legendre PRF originally proposed by Damgård in the PRG setting, namely the Jacobi PRF and the power residue PRF.

Keywords: Cryptanalysis · Legendre PRF · MPC-friendly primitives · Collision attack

*This is an updated version of the paper published in IACR Transactions on Symmetric Cryptology 2020(1) [BBUV20]. The work of Ward Beullens and Tim Beyne was supported by a PhD Fellowship from the Research Foundation - Flanders (FWO). The work of Aleksei Udovenko and Giuseppe Vitto was supported by the Luxembourg National Research Fund (FNR) project FinCrypt (C17/IS/11684537). The experiments presented in this paper were carried out using the HPC facilities of the University of Luxembourg [VBCG14] – see `https://hpc.uni.lu`.

1 Introduction

The Legendre symbol is a multiplicative function modulo an odd prime number p that assigns to an element $a \in \mathbb{F}_p$ the value $1, 0$ or -1 depending on whether or not a is a square. Specifically,

$$\left(\frac{a}{p}\right) = \begin{cases} 1 & \text{if } a = b^2 \text{ for some } b \in \mathbb{F}_p^\times \ , \\ 0 & \text{if } a = 0 \ , \\ -1 & \text{otherwise} \ . \end{cases}$$

The distribution of Legendre symbols has been a subject of study for number theorists at least since the early 1900s [Ala96, vS98, Jac06, Dav31, Dav39]. In particular, it follows from the Weil bound [Wei48] that the number of occurrences of a fixed pattern of l nonzero Legendre symbols among the integers $1, 2, \ldots, p-1$ modulo p is

$$\frac{p}{2^l} + \mathcal{O}(\sqrt{p}) \,,$$

as $p \to \infty$. In other words, the distribution of fixed length substrings of Legendre symbols converges to the uniform distribution.

In 1988, Damgård [Dam90] conjectured pseudorandom properties of the sequence

$$\left(\frac{k}{p}\right), \left(\frac{k+1}{p}\right), \left(\frac{k+2}{p}\right), \ldots,$$

where k has been sampled from $\mathbb{F}_p$ uniformly at random. He proposed to use this construction as a pseudorandom number generator. More recently, Grassi *et al.* [GRR$^+$16] have proposed the same construction as a candidate pseudorandom function and have shown that it can be evaluated very efficiently in the secure multiparty computation setting. Concretely, the *Legendre pseudorandom function* $L_k(x)$ is defined by mapping the Legendre symbol with a secret shift k to $\{0, 1\}$:

$$L_k(x) = \left\lfloor \frac{1}{2}\left(1 - \left(\frac{k+x}{p}\right)\right)\right\rfloor,$$

where p is a public prime number.

Damgård's work additionally considers several generalizations of the Legendre PRG that could be more efficient and/or more secure. One of these is to replace the Legendre symbols above by Jacobi symbols. In this case, the public modulus n is taken to be a product $\prod_i p_i$ of odd primes. Recall that the Jacobi symbol of $a \in \mathbb{F}_p$ is defined as

$$\left(\frac{a}{n}\right) = \prod_i \left(\frac{a}{p_i}\right).$$

Damgård argues that Jacobi symbols are more secure by showing that the Jacobi generator is strongly unpredictable if the Legendre generator is weakly unpredictable. Further, he notes that calculating Jacobi symbols is more efficient because computing them reduces to computing Legendre symbols modulo each of the smaller prime factors. A second generalization proposed by Damgård is the use of higher power residue symbols instead of quadratic residue symbols. Concretely, for a prime p with $p \equiv 1 \mod r$, he proposes to use the r-th power residue symbol $a \mapsto a^{(p-1)/r} \mod p$. This potentially increases the throughput of the PRF, because we now obtain $\log_2 r$ bits of output per PRF call rather than a single bit.

Very recently, the Legendre PRF was proposed to be used in the Ethereum 2.0 proof-of-custody mechanism [Fei19b]. In this context, several cryptanalysis bounties were announced by the Ethereum foundation during the CRYPTO 2019 rump session [Fei19a]. Among the proposed challenges, there are concrete instances of the Legendre PRF with expected security levels ranging from 44 to 128 bits of security. For each instance, 2^{20} sequential output bits are given and the goal is to recover the secret key.

Despite the longevity of Damgård's pseudorandomness conjecture and the recent surge of application-oriented interest in the Legendre PRF, relatively few cryptanalytic results are available. It is known that, given quantum query access to L_k, the key k can be recovered with a single query to L_k and in quantum polynomial time [vDH00]. No subexponential attacks are known in the classical setting or the setting where a quantum adversary is only allowed to query L_k classically.

The best cryptanalytic results in the classical setting are due to Khovratovich [Kho19], who gives a memoryless birthday-bound attack. His attack recovers the key with a computational cost of $\mathcal{O}(\sqrt{p} \log p)$ Legendre symbol evaluations when given $\sqrt{p} \log p$ queries to L_k. Khovratovich also considers a higher-degree variant of the Legendre PRF, where the output is computed as the Legendre symbol of a secret polynomial in the input. Similar to the Jacobi symbol generalization, the higher-degree Legendre PRF potentially offers security and efficiency benefits.

Contributions. This paper aims to advance the state-of-the-art in the cryptanalysis of the Legendre PRF by improving upon Khovratovich's attacks on the one hand, and by providing the first security analysis of the Jacobi and power residue symbol generalizations on the other hand. Table 1 provides a summary of our main results. The main improvement stems from the fact that, unlike earlier work, we manage to exploit the multiplicative property of the Legendre symbol. The practical relevance of our attacks is demonstrated by our solution of the first three concrete Legendre PRF challenges proposed by the Ethereum foundation [Fei19b]. These were expected to correspond to a security level of 44 and 54 bits, but our attacks imply that the actual security levels for these challenges are significantly lower.

After introducing the necessary preliminaries in Section 2, we show how the Khovratovich attack can be significantly improved in the low-data setting. In particular, for $M \leq \sqrt[4]{p}$ queries, the attack in Section 3 of this paper recovers the key with a time-complexity of $\mathcal{O}(p \log^2 p/M^2)$ Legendre symbol evaluations and a memory cost of $\mathcal{O}(M^2)$. In Section 4, the attack from Section 3 is generalized to the higher-degree case. As before, this amounts to a significant improvement in the low-data setting. In addition, for $d \geq 3$ and with $M = p$ queries, we gain a factor of p in time-complexity compared to Khovratovich's results. Furthermore, in Section 4, a large class of weak keys for the higher-degree Legendre PRF is shown to exist. For keys in this class, key-recovery requires roughly $\mathcal{O}(p^{\lceil d/2 \rceil} d \log p)$ operations with only $d\lceil \log p \rceil$ queries to the PRF. This attack requires $\mathcal{O}(p^{\lfloor d/2 \rfloor} d \log p)$ bits of memory, but trade-offs are available using Van Oorschot-Wiener golden collision search. We also give a reduction to the unique k-XOR problem, which results in further time-memory trade-offs.

The first of Damgård's generalizations is discussed in Section 6. Specifically, it will be shown that the Jacobi PRF can be broken with cost proportional to the cost of breaking the Legendre PRF for each of the prime factors of the modulus separately. The power residue symbol generalization is analyzed in Section 7. Besides the straightforward generalization of the attack from Section 3 to the r-th power residue symbol PRF, we additionally provide a more efficient attack for the case where r is large.

Finally, concrete implementation results are provided in Section 8. We report on the specific amount of time and memory that was necessary to solve the first three Legendre PRF challenges of the Ethereum foundation. These results showcase the practical relevance of our attacks.

Concurrent work. Days after this work first appeared on ePrint, Kaluđerović *et al.* [KKK20] solved the next Legendre PRF challenge. Their attack uses similar ideas to our attack, but with an improved complexity of $\mathcal{O}(M^2/\log p + p \log p \log\log p/M^2)$ operations on a machine with word size $\Theta(\log p)$.

Table 1: Query, time and memory requirements of previous and new attacks on the Legendre PRF. The reported time and memory values are asymptotic upper bounds ($\mathcal{O}$-notation) and assume a machine with word size $\Theta(\log p)$, ℓ and s denote the time-complexity of computing a Legendre and power residue symbol respectively. The attack strategy for composite moduli from Section 6 can be combined with any of the attacks in this table.

	Reference	Queries	Time	Memory
Legendre PRF	Randomized [Kho19]	$\log p$	$\ell p \log p$	$\log p$
	Khovratovich [Kho19]	$\sqrt{p} \log p$	$\ell \sqrt{p} \log p$	$\log p$
	Section 3.1	M	$M + \ell p \log p / M$	$M \log p$
	Section 3.3	M	$M^2 + \ell p \log^2 p / M^2$	M^2
	Section 3.4	M	$M^2 + p \log^2 p / M^2$	$M^2 / \log p$
Degree $d \geq 2$ Legendre PRF	Randomized [Kho19]	$\log p$	$\ell p^d \, d \log p$	$d \log p$
	Khovratovich [Kho19]	p	$\ell p^{d-1} d \log p$	$d \log p$
	Section 4	M	$M^2 + \ell p^d d^2 \log^2 p / M^2$	M^2
	Section 5	$d \log p$	$p^{\lceil d/2 \rceil} d \log p$	$p^{\lfloor d/2 \rfloor} d \log p$
r-th power-residue PRF	Section 7.2	M	$M^2 + sp \log^2 p / (M^2 \log^2 r)$	$M^2 \log r$
	Section 7.3	M	$M + sp \log^2 p / (Mr \log^2 r)$	$M \log r$

2 Preliminaries

After introducing the Legendre PRF and some related notation in Section 2.1, Section 2.2 recalls how Legendre and power residue symbols can be computed efficiently. Finally, Sections 2.3 and 2.4 discuss Khovratovich's attacks on the Legendre PRF and its higher degree variant.

2.1 Legendre PRF

Definition 1 (Legendre function)**.** For a given odd prime p, we consider the function

$$\begin{aligned} l : \mathbb{F}_p &\to \mathbb{F}_2 \\ x &\mapsto \left\lfloor \frac{1}{2} \left(1 - \left(\frac{x}{p} \right) \right) \right\rfloor \end{aligned}$$

which maps quadratic residues modulo p to $0 \in \mathbb{F}_2$ and quadratic non-residues to $1 \in \mathbb{F}_2$.

Definition 2 (Legendre PRF)**.** Let p be an odd prime and d a positive integer. The *degree d-Legendre PRF* over $\mathbb{F}_p$ is a family of functions $L_k : \mathbb{F}_p \to \mathbb{F}_2$ such that for each $k \in \mathbb{F}_p^d$,

$$L_k(x) = l\big(x^d + \textstyle\sum_{i=0}^{d-1} k_{i+1}\, x^i\big).$$

Remark 1. For any given field $\mathbb{F}_p$, the Legendre symbol is *multiplicative*, i.e.

$$\left(\frac{ab}{p}\right) = \left(\frac{a}{p}\right)\left(\frac{b}{p}\right) \quad \text{for all } a, b \in \mathbb{F}_p.$$

In terms of the Legendre function l, multiplication of inputs corresponds to addition in $\mathbb{F}_2$ of the respective images. Indeed

$$l(ab) = l(a) \oplus l(b) \quad \text{for all } a, b \in \mathbb{F}_p^\times,$$

where $\oplus$ denotes addition in $\mathbb{F}_2$.

In our analysis, we will often consider sequential evaluations of a given degree d Legendre PRF L_k starting from a point a with an additive or multiplicative step b. We call such vectors *L-sequences.*

Definition 3 (L-sequences)**.** Let p be an odd prime, m a positive integer and $a, b \in \mathbb{F}_p$. For a given L_k over $\mathbb{F}_p$, we define the *arithmetic L-sequence of length m with starting point a and stride b* as the $\mathbb{F}_2^m$-vector

$$L_k(a + b\,[m]) := (\, L_k(a), L_k(a+b), \ldots, L_k(a + (m-1)b)\,).$$

Similarly, we define the *geometric L-sequence of length m with starting point a and common ratio b* as the $\mathbb{F}_2^m$-vector

$$L_k(a \cdot b^{[m]}) := (L_k(a),\, L_k(a \cdot b),\, \ldots,\, L_k(a \cdot b^{m-1}\,)).$$

To justify the correctness of our attack, the following property of L_k will be assumed.

Assumption 1. *Let p be an odd prime and d a positive integer. Let $m = d\lceil \log p \rceil$. For all $k \in \mathbb{F}_p^d$, then as $p \to \infty$, there exist at most $\mathcal{O}(1)$ keys $k' \in \mathbb{F}_p^d$ such that $L_{k'}([m]) = L_k([m])$.*

2.2 Evaluating Legendre and Power Residue Symbols

Using the law of quadratic reciprocity, i.e. for odd coprime integers p and q

$$\left(\frac{p}{q}\right)\left(\frac{q}{p}\right) = (-1)^{\frac{p-1}{2}\frac{q-1}{2}},$$

Legendre symbols (and more generally Jacobi symbols) can be computed at essentially the same cost as a GCD computation. Using the Euclidean algorithm, the cost of a Legendre symbol computation is $\mathcal{O}(\log p)$ arithmetic operations, or $\mathcal{O}(\log^2 p \log\log p)$ bit operations. Brent and Zimmerman [BZ10] give an asymptotically better algorithm with complexity $\mathcal{O}(\log p \log^2 \log p)$. Power residue symbols can be computed via modular exponentiation in time $\mathcal{O}(\log p \log(p/r) \log\log p)$. In the remainder of this paper, we will often refer to the cost of an algorithm in terms of the number of Legendre symbol computations or power residue symbol computations.

2.3 Attacks on the Linear Legendre PRF

Khovratovich [Kho19] describes a chosen plaintext attack for the linear Legendre PRF L_k that recovers $k \in \mathbb{F}_p$ with $\mathcal{O}(\sqrt{p}\log p)$ queries to L_k. The attack is based on a memoryless collision search between two specific functions and can be briefly summarized as follows.

Let $m = \lceil \log p \rceil$ and consider the functions $x \mapsto L_k(x+[m])$ and $x \mapsto L_0(x+[m])$. Note that the L-sequence $L_k(x+[m])$ is available by querying the Legendre PRF, whereas $L_0(x+[m])$ does not depend on k. By Assumption 1, a collision between $x \mapsto L_k(x+[m])$ and $x \mapsto L_0(x+[m])$ yields k with high probability. Indeed, let $a, b \in \mathbb{F}_p$ be such that $L_k(a+[m]) = L_0(b+[m])$. We have

$$L_0(a+k+[m]) = L_0(b+[m]).$$

In accordance with Assumption 1, the number of superfluous candidates for k satisfying the above equality is expected to be at most $\mathcal{O}(1)$.

Collisions between $x \mapsto L_k(x+[m])$ and $x \mapsto L_0(x+[m])$ can be found with a generic memoryless collision search method [MOM92, vW94] in $\mathcal{O}(\sqrt{p})$ evaluations of both functions. Since computing each L-sequence requires $m = \mathcal{O}(\log p)$ calls to L_k, the overall complexity sums up to $\mathcal{O}(\sqrt{p}\log p)$ queries to L_k and L_0. More generally, if only M queries to L_k are allowed, a collision can be found with $\mathcal{O}(p \log^2 p/M)$ queries to L_0. This will be discussed in detail in Section 3.1.

We note that Khovratovich's original attack builds sequences of length m using arbitrary evaluations of the Legendre function L_k rather than consecutive ones. This difference does not affect the overall attack complexity, but by using L-sequences we will be able to reduce the data complexity in Section 3.

2.4 Attacks on the Higher-Degree Legendre PRF

Khovratovich [Kho19] also presents a generalization of the chosen plaintext attack from Section 2.3 to the quadratic case and, ultimately, to arbitrary degrees.

Let $k = (k_1, k_2) \in \mathbb{F}_p^2$ and consider the associated quadratic Legendre PRF L_k. Choose any $r \in \mathbb{F}_p^\times$. From the multiplicative property of the Legendre symbol we get that for any $a \in \mathbb{F}_p$ and $j \in \mathbb{Z}$,

$$L_{(r^{2j}\,k_1, r^j\,k_2)}(a) = l(r^{2j}) \oplus L_{(k_1,k_2)}(ar^{-j}) = L_k(ar^{-j}),^{1}$$

since r^{2j} is clearly a quadratic residue modulo p. Let $m = 2\lceil \log p \rceil$. If we find a $k' \in \mathbb{F}_p^2$ and a $j \in \mathbb{Z}$ such that

$$L_{k'}(r \cdot r^{[m]}) = L_k(r^{1-j} \cdot r^{[m]}),$$

then we successfully recover k by letting $k_1 = k_1' r^{-2j}$ and $k_2 = k_2' r^{-j}$. As for the linear case, such a collision can be found memorylessly with $\mathcal{O}(p)$ queries to L_k and $\mathcal{O}(p)$ Legendre symbol computations.

For the general case, consider the degree-d Legendre PRF L_k. Similarly to the quadratic case, we have for each $a \in \mathbb{F}_p$ and $j \in \mathbb{Z}$ that

$$L_{k_1 r^{dj}, k_2 r^{(d-1)j}, \ldots, k_d r^j}(a) = l(r^{dj}) \oplus L_k(ar^{-j}).$$

By guessing the coefficients $k_3, \ldots, k_d$, it is possible to attack the remaining coefficients k_1 and k_2 using geometric L-sequences of length $d\lceil \log p \rceil$ similar to the quadratic case. It follows that k can be recovered using $\mathcal{O}(p^{d-2} \cdot p \cdot d \log p) = \mathcal{O}(p^{d-1} d \log p)$ Legendre symbol evaluations, given $\mathcal{O}(p)$ queries to L_k.

3 Improved Attack on the Linear Legendre PRF

In this section, we show how Khovratovich's attack (Section 2.3) on the Legendre PRF can be improved when the total number of available queries is less than $\sqrt{p}$. Although, in its simplest form, our method requires additional memory, we discuss several techniques to reduce memory requirements while keeping the same overall time complexity.

[1]This equation, and many other equations in this paper, only holds if none of the involved Legendre symbols evaluate to zero. Since this does not pose a problem in practice we choose to ignore this issue for notational convenience.

3.1 Table-Based Collision Search

We first transform the attack by Khovratovich into a table-based collision search.

Let M be the allowed number of queries to the oracle L_k, where $\log p \ll M < \sqrt{p}$. Let $m = \lceil \log p \rceil$ and let $\tilde{M} = M - m + 1$. The attack proceeds as follows:

1. Store in a table $\mathcal{T}$ the pairs $(L_k(a + [m]), a)$ for all $a \in \{0, \ldots, \tilde{M} - 1\}$.
2. Sample b uniformly at random from $\mathbb{F}_p$ until $(L_0(b + [m]), a) \in \mathcal{T}$ for some $a \in \{0, \ldots, \tilde{M} - 1\}$. For each a corresponding to such a collision, a candidate key $\tilde{k}$ is recovered as $\tilde{k} = b - a$. By Assumption 1, the number of candidate keys is at most $\mathcal{O}(1)$. Candidate keys $\tilde{k}$ can be tested by comparing one or more entries of $\mathcal{T}$ with the corresponding arithmetic L-sequences with starting point $\tilde{k}$.

Regarding the time and memory complexity of this attack, we note that the first step requires M queries to L_k, from which we obtain $\tilde{M}$ arithmetic L-sequences that are stored using $\mathcal{O}(M \log p)$ memory. The second step requires $\mathcal{O}(p \log p / M)$ evaluations of the Legendre symbol and no additional memory is needed. Hence, the overall computational cost of the attack is $\mathcal{O}(M + p \log p / M)$.

Note that this variant of the attack already reduces the query and time complexities by a $\log p$ factor compared to the memoryless collision search, although a significant amount of memory is employed.

Remark 2. The above attack can be made deterministic by choosing $b \in \{0, \ldots, \lfloor p/\tilde{M} \rfloor\}$ and considering the sequences $v = L_0(b\tilde{M} + [m])$ in the second step of the attack. Indeed, it is easy to see that for any $k \in \mathbb{F}_p$, the arithmetic L-sequence at offset $\tilde{M}\left\lceil k/\tilde{M} \right\rceil$ will be computed in both steps of the attack and the correct key is guaranteed to be recovered after at most $\mathcal{O}(M + p \log p / M)$ Legendre symbol evaluations.

3.2 Expanding the Number of L-Sequences

We now show that the table can be expanded without increasing the number of queries M. The key idea is to exploit the *multiplicative* property of the Legendre symbol.

Lemma 1. *Let m be a positive integer and $k \in \mathbb{F}_p$. For any $b \in \mathbb{F}_p^{\times}$ and $a \in \mathbb{F}_p$*

it holds that

$$L_{k/b}(a/b + [m]) = (l(b), \dots, l(b)) \oplus L_k(a + b[m]).$$

Proof. Immediate by the multiplicative property of l. □

Lemma 2. *Let $k \in \mathbb{F}_p$ and $m \leq M$ positive integers. Then from the arithmetic L-sequence $L_k([M])$, it is possible to extract $\sim M^2/m$ arithmetic L-sequences of the form $L_{k/b}(a/b + [m])$ for distinct pairs $(a, b) \in \mathbb{F}_p \times \mathbb{F}_p^\times$.*

Proof. Let b a positive integer such that $b \leq \lfloor M/m \rfloor$. By Lemma 1, we get

$$L_k(a + b[m]) = (l(b), \dots, l(b)) \oplus L_{k/b}(a/b + [m])$$

for any $a \in [0, M - bm + 1)$, thus each b yields a total of $M - bm + 1$ L-sequences of length m. Moreover, since $L_k(a - b[m])$ is equal to the sequence $L_k(a - b(m-1) + b[m]) = L_k(a' + b[m])$ written in reverse order, we can consider negative values for b too, thus doubling the total number of sequences. Hence, the total number of arithmetic L-sequences of length m that can be extracted from $L_k([M])$ equals

$$2 \sum_{b=1}^{\lfloor M/m \rfloor} (M - bm + 1) \sim \frac{2M^2}{m} - m \sum_{b=1}^{M/m} b \sim \frac{2M^2}{m} - \frac{M^2}{m} = \frac{M^2}{m}. \qquad \square$$

3.3 An Improved Table-Based Collision Search

The observations from Section 3.2 will now be used to improve the table-based collision search from Section 3.1.

As before, let M be the allowed number of queries to the oracle L_k, where $\log p \ll M < \sqrt{p}$. Let $m = \lceil \log p \rceil$. The attack proceeds as follows:

1. Query the sequence $L_k([M])$ and extract $\sim M^2/m$ sequences of the form $L_{k/b}(a/b + [m])$ from it. This is possible by Lemma 2. Store all of the triples $(L_{k/b}(a/b + [m]), a, b)$ in a table $\mathcal{T}$.
2. Sample c uniformly at random from $\mathbb{F}_p$ until $(L_0(c + [m]), a, b) \in \mathcal{T}$ for some a and b. For each pair (a, b) corresponding to such a collision, a candidate key $\tilde{k}$ is recovered as $\tilde{k} = bc - a$. By Assumption 1, the number of candidate keys is at most $\mathcal{O}(1)$. As before, the correctness of candidate keys $\tilde{k}$ can easily be verified.

The first step of the attack requires M queries to L_k and $\sim M/m$ Legendre symbol evaluations. Storing the table $\mathcal{T}$ requires $\mathcal{O}(M^2)$ memory. In the second phase, an average of $\sim mp/M^2$ samples must be tested before a collision is found. Hence, the computational cost of this step is dominated by $\mathcal{O}(pm^2/M^2)$ Legendre symbol evaluations.

It follows that the overall cost of the attack is dominated by the extraction of $\mathcal{O}(M^2/m)$ sequences, the evaluation of $\mathcal{O}(M/m + p\log^2 p/M^2)$ Legendre symbols and a memory requirement of $\mathcal{O}(M^2)$. For $M < \sqrt{p}$, this is always an improvement over the attack from Section 3.1 – possibly after discarding some of the data.

3.4 Additional Optimizations

This section describes a number of additional optimizations that allow a further reduction of both the time and the memory complexity of the attack by a factor $\Omega(\log p)$.

Using Consecutive Values of c

The second step of the attack from Section 3.3 can be optimized by choosing consecutive values of c rather than uniform random samples. This approach allows us to reuse most of the Legendre symbol computations since, for example, $L_0(c + [m])$ and $L_0(c + 1 + [m])$ overlap almost completely. A priori, this allows reducing the number of Legendre symbol computations by a factor of $\Omega(m)$. However, there is an important caveat: since the guesses for c are not independent, the expected number of iterations of the second step is no longer pm/M^2. To see why this is the case, recall that for any c, the algorithm will output the correct key k if there exists $(*, a, b) \in \mathcal{T}$ such that $k = bc - a$. Since the table contains an entry $(*, a, b)$ for all sufficiently small values of a and b, it is clear that if the table contains $(*, a, b)$ such that $k = bc - a$ it is likely to also contain $(*, a' = a + b, b)$ such that $k = b(c + 1) - a'$. Therefore, if c is a good guess, then $c + 1$ is also likely to be a good guess. Since the "good" values of c are clustered together in groups of size $\mathcal{O}(m)$, we expect the required number of iterations to be $\mathcal{O}(pm^2/M^2)$, which means that the factor $\Omega(m)$ that we saved by using consecutive guesses for c is lost again. However, we can still use this idea to reduce the memory complexity of the algorithm: by only storing one entry $(*, a, b)$ for each cluster of good c's, i.e. we only store the triples $(*, a, b)$ such that $|a| < |b|$, the size of the table can be reduced by a factor of $\Omega(m)$

without impacting the time complexity of the attack.

Expanding the Number of L-Sequences in the Second Step

The idea outlined in Section 3.2 can be used to create new L-sequences from those computed during the second step of the attack. Indeed, after computing a large number of $w = \Omega(m)$ consecutive Legendre symbols $L_0(c + [w])$, it is possible to extract $\Omega(w^2/m^2)$ arithmetic subsequences of the form $L_0(c + c' + d[m])$ such that $|c'| < |d|$, with no need to compute additional Legendre symbols. Using the property that

$$L_0(c + c' + d[m]) = L_0((c + c')/d + [m]) \oplus L_0(d)$$

we can then do $\Omega(w^2/m^2)$ table lookups. Asymptotically, this allows to amortize away the cost of computing Legendre symbols, so the time complexity is dominated by the extraction of $\mathcal{O}(pm^2/M^2)$ subsequences rather than by the computation of $\mathcal{O}(pm^2/M^2)$ Legendre symbols.

Not Storing Reverse Sequences

Since the sequence $a + b[m]$ is just the reverse of the sequence $a + b(m-1) - b[m]$, there is some redundancy in the lookup table. Indeed, for each entry $(s, a, b) \in \mathcal{T}$, the reverse sequence corresponding to the entry $(s', a + b(m-1), -b)$ is also stored. If, instead, we only store either the sequence or its reverse (e.g. by storing the lexicographically smallest sequence), then the memory requirements are reduced by a factor of two without affecting the overall time-complexity just by looking up either the sequence $L_0(c + [m])$ *or* its reverse in $\mathcal{T}$, depending which comes first lexicographically.

4 Application to the Higher-Degree Legendre PRF

In this section we generalize the attack described in Section 3 to Legendre PRFs of degree $d > 1$. In Section 4.1 it is shown how to expand the number of L-sequences in the higher-degree setting. The resulting attack is detailed in Section 4.2.

4.1 Expanding the Number of L-Sequences

In order to generalize Lemma 2, we need to extend Lemma 1 to the higher-degree case. This is the object of Lemma 3.

Lemma 3. *For any positive integer m, $b \in \mathbb{F}_p^\times$ and $a \in \mathbb{F}_p$, there exists an invertible affine transformation $T_{a,b}$ such that for any $k \in \mathbb{F}_p^d$,*

$$L_{T_{a,b}(k)}([m]) = (l(b^d), \dots, l(b^d)) \oplus L_k(a + b[m]).$$

Moreover, for any choice of $(a,b) \in \mathbb{F}_p \times \mathbb{F}_p^\times$, the transformation $T_{a,b}$ can be efficiently computed.

Proof. Lef f be the monic degree d polynomial with coefficient vector k, and let $T_{a,b}(k)$ be the coefficient vector of the monic polynomial $f(a+bx)/b^d$. Then, by the multiplicative property of the Legendre symbol, we have that

$$L_{T_{a,b}(k)}([m]) = (l(b^d), \dots, l(b^d)) \oplus L_k(a + b[m]).$$

Furthermore, it is not hard to see that $T_{a,b}$ is invertible, affine and that it can be computed efficiently. □

Lemma 4. *Let $k \in \mathbb{F}_p^d$ and $m \leq M$ positive integers. Then from the arithmetic L-sequence $L_k([M])$, it is possible to extract $\sim M^2/m$ arithmetic L-sequences of the form $L_{k'}([m])$ with k' as defined in Lemma 3 for distinct pairs $(a,b) \in \mathbb{F}_p \times \mathbb{F}_p^\times$.*

Proof. The proof is completely analogous to that of Lemma 2. □

4.2 An Improved Table-Based Collision Search

The attack proceeds in essentially the same way as described in Section 3.3 for the linear case. Let M be the allowed number of consecutive queries to the oracle L_k. Let $m = d \lceil \log p \rceil$. The attack comprises the following steps:

1. Query the sequence $L_k([M])$ and extract $\sim M^2/m$ sequences of the form $L_{k'}([m])$ from it. This is possible by Lemma 4. Store all of the triples $(L_{k'}([m]), a, b)$ in a table $\mathcal{T}$.

2. Sample k' uniformly at random from $\mathbb{F}_p^d$ until $(L_{k'}([m]), a, b) \in \mathcal{T}$ for some a and b. For each pair (a, b) corresponding to such a collision, a candidate key $\tilde{k}$ can be recovered from k, a and b as in Lemma 3. By Assumption 1, the number of candidate keys is at most $\mathcal{O}(1)$. As before, the correctness of candidate keys can easily be verified.

As in Section 3.3, the computational cost of the first step is dominated by the extraction of $\mathcal{O}(M^2/m)$ sequences. For the second step, at most $\mathcal{O}(p^d\, m^2/M^2)$ Legendre symbols are expected to be evaluated. Hence, the total computational cost of the attack consists of $\mathcal{O}(M^2/m)$ sequence extractions and $\mathcal{O}(p^d\, d^2 \log^2 p/M^2)$ Legendre symbol evaluations. The attack requires $\mathcal{O}(M^2)$ memory.

For $d \geq 3$, the time-complexity is minimized for $M = p$. The time complexity is then $\mathcal{O}(p^{d-2} d^2 \log^2 p)$ Legendre symbol computations. Hence, we gain a factor of p in time relative to the attacks by Khovratovich [Kho19].

5 Weak Keys in the Higher-Degree Legendre PRF

In this section, we exhibit a large class of weak keys for the higher-degree Legendre PRF. Our attacks are based on the observation that for some keys, the corresponding monic polynomial factors as a product of polynomials of lower degree.

5.1 A Birthday-Bound Attack for Some Keys

Consider the Legendre PRF of degree $d \geq 2$ over $\mathbb{F}_p$ for a prime p. Recall that the key $k \in \mathbb{F}_p^d$ of the PRF corresponds to the monic polynomial $f(x) = x^d + \sum_{i=0}^{d-1} k_{i+1} x^i \in \mathbb{F}_p[x]$. The attack in this section is based on the observation that, with high probability, the polynomial f has a factor of degree $t = \lfloor d/2 \rfloor$. In this case, there exist two monic polynomials $g, h \in \mathbb{F}_p[x]$ with $\deg g = t$ and $\deg h = d - t$ such that $f = gh$.

Assume that we are given the outputs of the PRF on $m = d\lceil \log p \rceil$ arbitrary inputs, for example the sequence $L_k([m])$. Then, by the multiplicative property of the Legendre symbol[2],

$$L_k([m]) = l(g([m])) \oplus l(h([m])).$$

Hence, the problem of finding the secret key $k \in \mathbb{F}_p^d$ reduces to a simple collision search:

1. Query the sequence $L_k([m])$ from the PRF. For each monic polynomial g of degree t, store the pair $(L_k([m]) \oplus l(g([m])), g)$ in a table $\mathcal{T}$.
2. Sample monic polynomials h of degree $d - t$ until $(l(f([m])), g) \in \mathcal{T}$ for some monic polynomial g of degree t. For each such g, recover a candidate key from the coefficients of gh. By Assumption 1, the number of candidate keys will be at most $\mathcal{O}(1)$.

For $t = \lfloor d/2 \rfloor$, this attack requires $\mathcal{O}(p^{\lfloor d/2 \rfloor} d \log p)$ bits of memory and its time complexity is dominated by $\mathcal{O}(p^{\lceil d/2 \rceil} d \log p)$ operations. The attack requires only $m = \mathcal{O}(d \log p)$ queries to the PRF. We use the fact that all Legendre symbols modulo p can be precomputed in $O(p)$ operations.

Using Van Oorschot-Wiener golden collision search [vW94], an improved time-memory trade-off can be obtained: given M bits of memory, the key can be recovered with a time-complexity of $\mathcal{O}(d \log p \sqrt{p^{3d/2}/M})$ Legendre symbol evaluations.

Even if the polynomial f does not have a factor of degree exactly $\lfloor d/2 \rfloor$, it might still have a factor of large degree $t < \lfloor d/2 \rfloor$. In this case, the same strategy results in an attack with time complexity $\mathcal{O}(p^{d-t} d \log p)$ and memory complexity $\mathcal{O}(p^t d \log p)$. This gives a trade-off between more efficient attacks on a smaller fraction of keys (when t is large) or less efficient attacks on a larger fraction of the keys (when t is small). This trade-off is illustrated in Figure 1. The figure shows the time-complexity of the attack for a desired fraction of attackable keys. The construction of Figure 1 is based on the following fact [Tao15]: the fraction of monic degree-d polynomials whose factorization has exactly c_i monic irreducible factors of degree i is $1/\prod_{i=1}^{d} c_i!\, i^{c_i}$ as $p \to \infty$. By summing these probabilities over all integer partitions of d that allow a $(t, d-t)$ split, we obtain the probability that a uniformly random key is weak.

We conclude that if the key is chosen uniformly at random, the higher-degree Legendre PRF has security only up to the birthday bound. To completely

[2]For convenience, we extend our notation for arithmetic L-sequences (Definition 3) to arbitrary functions on $\mathbb{F}_p$. In particular, $l(g([m])) = (l(g(0)), \dots, l(g(m-1)))$.

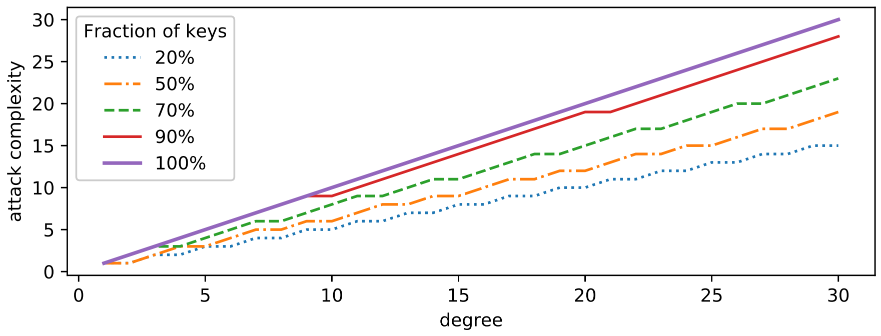


Figure 1: The complexity of the attack, measured as a power of p, as a function of the degree of f and the desired fraction of keys we want to attack.

prevent this class of attacks, one can choose the key k such that the corresponding polynomial f is irreducible.

5.2 Reduction to the Unique k-XOR Problem

More generally, the secret polynomial could factor into k polynomials of degree roughly d/k. For example, if d is divisible by k and $f = \prod_{i=1}^{k} f_i$ with $\deg f_i = d/k$, we have

$$L_k([m]) = \bigoplus_{i=1}^{k} l(f_i([m])).$$

That is, it suffices to find a solution to a variant of the k-XOR problem. Specifically, since each list has length $p^{d/k}$, a unique solution is expected. This makes Wagner's approach [Wag02] inapplicable, but some improvements over the attack in Section 5.1 are nevertheless possible.

In particular, for $k = 4$, the algorithm of Chose, Joux and Mitton [CJM02] leads to a time complexity $\tilde{\mathcal{O}}(p^{d/2})$ with only $\tilde{\mathcal{O}}(p^{d/4})$ memory. Corresponding time-memory trade-offs can also be obtained.

Finally, we mention that there exist asymptotically better quantum algorithms for the unique k-XOR problem. Bernstein *et al.* [BJLM13] give an $\widetilde{\mathcal{O}}(p^{0.3d})$ algorithm requiring $\widetilde{\mathcal{O}}(p^{0.2n})$ quantum-accessible quantum memory for $k = 4$. For any $k \geq 3$, Naya-Plasencia and Schrottenloher [NPS19] give algorithms running in time $\widetilde{\mathcal{O}}(p^{\beta_k d})$ where $\beta_k = (k + \lceil k/5 \rceil)/(4k)$ using $\widetilde{\mathcal{O}}(p^{0.2n})$ quantum-accessible quantum memory. For $k = 3$, there is an algorithm using $\widetilde{\mathcal{O}}(p^{d/3})$ time and $\widetilde{\mathcal{O}}(p^{d/3})$ quantum-accessible *classical* memory.

6 Jacobi Symbol PRF

The Jacobi pseudorandom generator was proposed by Damgård [Dam90] as a variation on the Legendre PRG. As discussed by Damgård [Dam90, §5], it is potentially more efficient because it can be computed as the exclusive-or of several Legendre PRGs with a relatively small modulus. In addition, Damgård showed that if the Legendre generator is weakly unpredictable, then the Jacobi generator is strongly unpredictable. A generator is defined to be weakly unpredictable if, for all polynomials f, there exist only finitely many integers $m \geq 0$ such that the next output bit in a sequence of length m can be predicted with probability greater than $1 - 1/f(m)$. Similarly, the generator is said to be strongly unpredictable if the probability of successful prediction exceeds $1/2 + 1/f(m)$ for only finitely many m. For a more formal definition, see [Dam90, §3] and references therein.

This section investigates the security of the Jacobi PRF in the chosen-plaintext setting. Whereas the unpredictability result of Damgård could be regarded as a positive result related to the security of the Jacobi PRF, it remains inconclusive concerning its concrete security. Indeed, strong unpredictability is a weaker property than PRF-security and, in addition, it is only an asymptotic notion of security.

Clearly, the cost of a key-recovery attack on the Jacobi PRF is at least the cost of attacking a Legendre PRF corresponding to a prime factor of the modulus. Below, a chosen-plaintext key-recovery attack on the Jacobi PRF is given which nearly attains this lower bound. Hence, for most purposes, the Jacobi PRF offers little benefit over the Legendre PRF.

Let $n = \prod_{i=1}^{m} p_i$ with $p_1, \ldots, p_m$ distinct odd primes. Note that it may be assumed that the prime factors of n are distinct, since

$$\left(\frac{x+k}{n}\right) = \left(\frac{x+k}{\prod_{i=1}^{m} p_i^{e_i}}\right) = \prod_{\substack{i=1 \\ e_i \text{ odd}}}^{m} \left(\frac{x+k}{p_i}\right).$$

Let $\lambda_j = \prod_{\substack{i=1\\ i\neq j}}^{m} p_i$ and denote the inverse of λ_j modulo p_j by λ_j'. Then

$$\left(\frac{\lambda_j\, x + k}{n}\right) = \prod_{i=1}^{m} \left(\frac{\lambda_j\, x + k}{p_i}\right) = \left(\frac{\lambda_j}{p_j}\right)\left(\frac{k}{n/p_j}\right)\left(\frac{x + \lambda_j'\, k}{p_j}\right).$$

Hence, in the chosen-plaintext setting, the key-recovery attack on the Legendre PRF from Section 3 can be used to recover the key modulo p_j. The factor $\left(\frac{k}{n/p_j}\right)$ is not known to the attacker, but it is constant so the cost of the attack is increased by a factor of at most two. Given the value of the key modulo each prime factor of n, the Chinese remainder theorem yields the value of the key modulo n. Hence, key recovery for the Jacobi symbol costs at most $\mathcal{O}(mM^2 + \sum_{i=1}^{m} p_i \log^2 p_i / M^2)$ Legendre symbol evaluations. The same strategy is applicable to the higher-degree case and can also be combined with the attacks in Section 7 below. Note that a distinguishing attack on the Jacobi PRF reduces to a distinguishing attack on the Legendre PRF corresponding to the smallest prime factor of the modulus.

7 Attacks on the Power Residue PRF

The MPC protocol of Grassi *et al.* [GRR$^+$16] for computing the Legendre PRF requires only three rounds of communication, which makes the Legendre PRF superior among the PRF constructions investigated by Grassi *et al.* in terms of latency. However, since the Legendre PRF only produces one bit of output, it compares less favorably in terms of throughput than e.g. MiMC [AGR$^+$16], a block cipher that outputs full field elements.

To mitigate this limitation of the Legendre PRF we can, as proposed by Damgård [Dam90], consider higher power residue symbols rather than quadratic residue symbols. If r divides $p-1$, the *r-th power residue symbol* of $x \in \mathbb{F}_p$ is defined as

$$\left(\frac{x}{p}\right)_r := x^{\frac{p-1}{r}} \mod p.$$

Jointly computing r-th power residue symbols in the MPC setting can be done at essentially the same cost as computing Legendre symbols with the advantage that $\log r$ bit outputs are produced instead. Therefore, this modification has the potential to significantly increase the throughput of the Legendre PRF at essentially no cost – keeping in mind that r should not be too large, since the corresponding power residue PRF might lose its security (e.g. $r = p - 1$). In this section we provide the first security analysis of the power residue PRF. We show that there exists an attack with time complexity $\mathcal{O}(p \log^2 p/(Mr \log^2 r))$, given $M \leq \sqrt{p}$ queries to the PRF.

7.1 Power Residue PRF

By generalising the Legendre function and the Legendre PRF to higher power residues, we obtain the following definitions:

Definition 4 (r-th power residue function)**.** Let p be a prime congruent to 1 mod r and g a generator of $\mathbb{F}_p^\times$. Then we define the r-th power residue function $l^{(r)} : \mathbb{F}_p \to \mathbb{Z}_r$ as

$$l^{(r)}(a) = \begin{cases} k \text{ if } a \not\equiv 0 \mod p \text{ and } a/g^k \text{ is an } r\text{-th power mod } p \\ 0 \text{ if } a \equiv 0 \mod p \end{cases}$$

Definition 5 (r-th power residue PRF)**.** Let p be a prime congruent to 1 modulo r. The *power residue PRF* over $\mathbb{F}_p$ is a family of functions $L_k^{(r)} : \mathbb{F}_p \to \mathbb{Z}_r$ such that for each $k \in \mathbb{F}_p$,

$$L_k^{(r)}(x) = l^{(r)}(k + x).$$

7.2 Generalising our Attack to the Power Residue PRF

The attacks described in Section 3 and Section 4 do not use any properties of the Legendre symbol other than its multiplicativity. Therefore, they trivially generalize to any multiplicative function with a hidden shift, including the r-th power residue function.

Unlike the quadratic case, the r-th power residue function can take r distinct values, so it suffices to consider L-sequences of length $\log p/\log r$. It follows that a straightforward generalization of our attack to r-th power residue Legendre PRFs requires $\mathcal{O}(p \log^2 p/(M^2 \log^2 r))$ power residue symbol evaluations and $\mathcal{O}(M^2 \log r)$ memory. However, for large values of r, there exists a better attack which is detailed in the next section.

7.3 Attacks for Large r

We first describe a very simple attack on the linear r-th power residue Legendre PRF that requires $\mathcal{O}(p/r)$ power residue symbol evaluations. In the following, denote the subgroup of $(p-1)/r$-th roots of unity of $\mathbb{F}_p^\times$ by $\mathbb{G}$. That is,

$$\mathbb{G} = \{x \in \mathbb{F}_p^\times \mid x^{(p-1)/r} = 1\}.$$

Remark that $\mathbb{G}$ is generated by g^r, where g is any generator of $\mathbb{F}_p^\times$.

By querying $L_k^{(r)}(0)$, the attacker immediately learns $l^{(r)}(k)$, the power residue symbol of $k \in \mathbb{F}_p$. We observe that this single query already narrows down the set of possible values for k to at most $(p-1)/r$ elements of $\mathbb{F}_p$. Indeed, from Definition 4, k is contained in the coset $g^s\mathbb{G}$, where g is any generator of $\mathbb{F}_p^\times$ and s is equal to $l^{(r)}(k)$. Therefore, an attacker can just go through all of these elements and check each candidate. Since, on average, only $\mathcal{O}(1)$ power residue symbols must be computed to check the validity of a candidate key, the attack requires $\mathcal{O}(p/r)$ power residue symbols evaluations. The attack requires a generator g, which can be precomputed in probabilistic subexponential time by factoring $p-1$.

We now explain a more general attack that requires $\mathcal{O}(p\log^2 p/(Mr\log^2 r))$ power residue symbol evaluations and $\mathcal{O}(M\log r)$ memory. The attack is similar to the table-based collision search from Section 3.1. A speed-up of a factor r is obtained by querying the PRF at more carefully chosen arithmetic L-sequences. Let $m = \lceil \log p/\log r \rceil$ and $M < p/r$. The attack proceeds as follows:

1. For M/m distinct values $a \in \mathbb{G}$, store each pair $(L_k^{(r)}(a[m]), a)$ in a table $\mathcal{T}$. Furthermore, query the PRF to get the value $s = L_k^{(r)}(0)$.
2. Sample x uniformly at random from the coset $g^s\mathbb{G}$ until $(L_0^{(r)}(x+[m]), a) \in \mathcal{T}$ for some value a. For each entry $(L_0^{(r)}(x+[m]), a) \in \mathcal{T}$ corresponding to such a collision, a candidate key is recovered as $\tilde{k} = xa$. By a variant of Assumption 1, the number of such candidate keys will be at most $\mathcal{O}(1)$.

The first step of the above attack uses $M = m \cdot (M/m)$ queries to $L_k^{(r)}$ and needs $\mathcal{O}(M\log r)$ memory to store the table $\mathcal{T}$. The key k is found when, in the second step, the attacker samples an x such that k/x is one of the a-values stored in the table. On average, $|\mathbb{G}|/(M/m) = \mathcal{O}(pm/(Mr))$ iterations of the second step are required in order to find a candidate key. Since each iteration requires m power residue symbol computations to evaluate $L_0^{(r)}(x + [m])$, it follows that the total time-complexity of the attack consists of $\mathcal{O}(M)$ storage operations and $\mathcal{O}(pm^2/(Mr)) = \mathcal{O}(p \log^2 p/(Mr \log^2 r))$ power residue symbol evaluations.

8 Implementation Results

This section discusses several aspects of our implementation of the attack from Section 3.3 that we applied to the key recovery puzzles proposed by the Ethereum foundation [Fei19b]. Using the attack from Section 3, we managed to solve three out of six challenges (including the test instance with a 40-bit prime). A summary of the instance parameters and the time and memory requirements of the attack is given in Table 2.

The source code of our implementation is publicly available at

`https://github.com/cryptolu/LegendrePRF`

Table 2: Parameters of the concrete challenges proposed by the Ethereum foundation [Fei19b]. For all instances, the first $M = 2^{20}$ consecutive PRF outputs were given. For the first three instances, the running time and peak memory usage is given, for the three hardest instances an estimation of time is provided (marked by †). All experiments were performed on a Dell C6420 server with two Intel Xeon Gold 6132 CPUs clocked at 2.6 GHz and 128 GB of RAM.

p	Security level[3] (bits)	Time (core-hours)	Memory / thread (GB)	Key
$2^{40} - 87$	20	< 0.001	< 1	`4e2dea1f3c`
$2^{64} - 59$	44	1.5	3	`90644c931a3fba5`
$2^{74} - 35$	54	1500	3	`384f17db02976dcf63d`
$2^{84} - 35$	64	2^{21}†	3	
$2^{100} - 15$	80	2^{37}†	3	
$2^{148} - 167$	128	2^{65}†	3	

[3]Expected security level (conservative estimate) prior to this work.

We compiled our C++ implementation of the attack using Clang 6.0.0 and executed it on a Dell C6420 server with two Intel Xeon Gold 6132 CPUs clocked at 2.6 GHz (28 cores) and 128 GB of RAM. The optimizations described in Section 3.4 allow to significantly reduce the required memory and the number of evaluations of the Legendre symbol. As a result, the table lookups are the bottleneck in our implementation. On average, a single thread required $0.08\,\mu s$ to compute and check a single 64-bit sequence. As discussed below, we expect to compute $p/2^{28}$ sequences on average before the key is recovered. Hence, the required core time to solve a challenge with a prime p and 2^{20} bits of PRF output can be estimated as $p/2^{28} \times 0.08\,\mu s$. The required memory is 1 GB per server and an additional 3 GB per thread. The parameters can be modified to reduce the memory without significantly decreasing the performance.

For the first three instances we successfully recovered the secret key of the PRF in a timespan close to our estimation. The corresponding keys are given in Table 2. The third instance was solved in under two hours using a cluster of 40 nodes with the described configuration. Further details about the main steps of the attack are provided below.

Step 1: Processing the PRF Output

As a first step we compute the set $\mathcal{T}$ consisting of all arithmetic sequences extracted from the sequence $L_k([2^{20}])$ given in the challenge. We chose to store sequences of length $m = 64$ since this length provides an acceptable rate of false-positives and enables to efficiently process sequences as 64-bit words. As a result, the set $\mathcal{T}$ contains approximately $M^2/(2m^2) = 2^{27}$ of such words-sequences.

A straightforward way to implement a set is by using a hash table, which has a constant amortized time-complexity for membership testing. However, this constant time may be quite large in practice, especially in the case of large tables. Random memory accesses are often the main bottleneck. In our case, the set $\mathcal{T}$ is never modified after its creation. To exploit this fact, we sort the elements of $\mathcal{T}$ and we store them in an array. Then, we compute membership queries in batches. First, we collect a large number of membership queries and we sort them. Then, we scan through the two sorted arrays checking for collisions. The bottleneck in this approach is represented by the sorting step of each batch of membership queries. The described set $\mathcal{T}$ contains 2^{27} 64-bit words and the corresponding sorted array requires 1 GB of memory. An extra 1 GB of memory is used to store information required for the key recovery. Note that the set $\mathcal{T}$ and the extra information are shared among all threads that are used to parallelize the workload of the next step.

Step 2: Random Sampling

The second and main step of the attack consists of sampling sequences $L_0(c+[m])$ for randomly chosen c and checking if they collide with an entry of $\mathcal{T}$. Note that the reversed sequence $L_0(c+[m])$ is checked if it is lexicographically smaller.

For a uniformly chosen $c \in \mathbb{F}_p$ we compute a long sequence $L_0(c+[t])$ and we extract a large number of m-bit sequences from it. More precisely, for all $b \in \{1, 2, \dots, 2^8\}$ and $a \in \{0, 1, \dots, t-1-b(m-1)\}$, we extract $L_0(c+a+b[m])$. The upper-bound for b is chosen as 2^8 since it is enough to make the time spent on computing Legendre symbols negligible. Furthermore, all these sequences can be computed on the fly by storing only the last sequence per pair (b, a). Indeed, for a large enough $i \in \mathbb{Z}$, after expanding the computed sequence $L_0(c+[i-1])$ by one Legendre symbol $L_0(c+i)$ we obtain a new sequence $L_0(c+i-b(m-1)+b[m])$ for each b. In other words, we obtain 2^8 sequences from each single consequent Legendre symbol computation.

As described above, the computed sequences are accumulated and checked in batches for a collision with the set $\mathcal{T}$. Each batch is sorted using base-2^8 radix sort and collisions are checked using a linear scan through the sorted batch and the sorted array of $\mathcal{T}$. In the case of a collision, a key candidate is recovered and checked against extra bits from the given PRF output.

Note that this step can be efficiently parallelized. Each thread starts with a uniformly random $a \in \mathbb{F}_p$ and proceeds as described above. After a predetermined amount of steps, a new value for a can be chosen to ensure a sufficiently uniform coverage of the possible offsets of the sequences.

9 Conclusions

In Section 3, a new attack on the Legendre PRF was presented. It is of particular interest in the low-data setting. Specifically, given $M \leq \sqrt[4]{p}$ queries, our attack recovers the key using $\mathcal{O}(p \log^2 p / M^2)$ Legendre symbol evaluations. The practical relevance of this result was demonstrated by solving the first two Legendre PRF challenges set out by the Ethereum foundation [Fei19b]. Several aspects of our implementation of the attack were discussed in Section 8.

In Section 4, it was shown how the technique from Section 3 yields improved attacks on the higher-degree generalization of the Legendre PRF. Further attacks on the higher-degree case were given in Section 5, where a large class of weak keys was revealed. Keys from this class can be recovered using $\mathcal{O}(p^{\lceil d/2 \rceil} d \log p)$ operations and $\mathcal{O}(p^{\lfloor d/2 \rfloor} d \log p)$ bits of memory. Further improvements to the memory usage, based on a reduction to the unique k-XOR problem, were also discussed. These weak key attacks can be prevented by choosing the key such that the corresponding monic polynomial is irreducible.

In addition to the above, we provided the first security analysis of the Jacobi and power-residue generalizations of the Legendre PRF. These extensions were first suggested – for the Legendre pseudorandom generator – at CRYPTO 1988 by Damgård [Dam90]. It was demonstrated in Section 6 that the key of a Jacobi PRF can be recovered with time-complexity proportional to the time-complexity of key-recovery on the Legendre PRF for each of the prime factors of the modulus separately. This result eliminates the potential efficiency benefits offered by Jacobi symbols.

Power residue symbols were considered in Section 7. The low-data attack from Section 3 equally applies in this setting, but we provide an additional attack that preforms better for large power residue symbols. Specifically, for r-th power residue symbols and given $M \leq \sqrt{p}$ queries, our key-recovery attack requires $\mathcal{O}(p \log^2 p/(rM \log^2 r))$ power residue evaluations and $\mathcal{O}(M)$ memory.

References

[AGR+16] Martin R. Albrecht, Lorenzo Grassi, Christian Rechberger, Arnab Roy, and Tyge Tiessen. MiMC: Efficient encryption and cryptographic hashing with minimal multiplicative complexity. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 191–219. Springer, Heidelberg, December 2016.

[Ala96] N. Aladov. Sur la distribution des résidus quadratiques et non-quadratiques d'un nombre premier p dans la suite $1, 2, \ldots, p-1$. *Matematicheskii Sbornik*, 18(1):61–75, 1896.

[BBUV20] Ward Beullens, Tim Beyne, Aleksei Udovenko, and Giuseppe Vitto. Cryptanalysis of the Legendre PRF and generalizations. *IACR Trans. Symm. Cryptol.*, 2020(1):313–330, 2020.

[BJLM13] Daniel J Bernstein, Stacey Jeffery, Tanja Lange, and Alexander Meurer. Quantum algorithms for the subset-sum problem. In

International Workshop on Post-Quantum Cryptography, pages 16–33. Springer, 2013.

[BZ10] Richard P Brent and Paul Zimmermann. An $\mathcal{O}(M(n)\log n)$ algorithm for the Jacobi symbol. In *International Algorithmic Number Theory Symposium*, pages 83–95. Springer, 2010.

[CJM02] Philippe Chose, Antoine Joux, and Michel Mitton. Fast correlation attacks: An algorithmic point of view. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 209–221. Springer, Heidelberg, April / May 2002.

[Dam90] Ivan Damgård. On the randomness of Legendre and Jacobi sequences. In Shafi Goldwasser, editor, *CRYPTO'88*, volume 403 of *LNCS*, pages 163–172. Springer, Heidelberg, August 1990.

[Dav31] Harold Davenport. On the distribution of quadratic residues (mod p). *Journal of the London Mathematical Society*, 1(1):49–54, 1931.

[Dav39] Harold Davenport. On character sums in finite fields. *Acta Mathematica*, 71(1):99–121, 1939.

[Fei19a] Dankrad Feist. Cryptanalyzing the Legendre PRF. CRYPTO rump session talk, August 2019.

[Fei19b] Dankrad Feist. Legendre pseudo-random function. `https://legendreprf.org`, 2019. Accessed: 2019-11-18.

[GRR$^+$16] Lorenzo Grassi, Christian Rechberger, Dragos Rotaru, Peter Scholl, and Nigel P. Smart. MPC-friendly symmetric key primitives. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 430–443. ACM Press, October 2016.

[Jac06] Ernst Erich Jacobsthal. *Anwendungen einer Formel aus der Theorie der quadratischen Reste*. PhD thesis, Friedrich-Wilhelms Universität zu Berlin, 1906.

[Kho19] Dmitry Khovratovich. Key recovery attacks on the Legendre PRFs within the birthday bound. Cryptology ePrint Archive, Report 2019/862, 2019. `https://eprint.iacr.org/2019/862`.

[KKK20] Novak Kaluđerović, Thorsten Kleinjung, and Dušan Kostić. Improved key recovery on the legendre prf. Cryptology ePrint Archive, Report 2020/098, 2020. `https://eprint.iacr.org/2020/098`.

[MOM92] Hikaru Morita, Kazuo Ohta, and Shoji Miyaguchi. A switching closure test to analyze cryptosystems. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 183–193. Springer, Heidelberg, August 1992.

[NPS19] María Naya-Plasencia and André Schrottenloher. Optimal merging in quantum k-xor and k-sum algorithms. Cryptology ePrint Archive, Report 2019/501, 2019. `https://eprint.iacr.org/2019/501`.

[Tao15] Terence Tao. Cycles of a random permutation and irreducible factors of a random polynomial. `https://terrytao.wordpress.com/2015/07/15/cycles-of-a-random-permutation-and-irreducible-factors-of-a-random-polynomial/`, 2015. Accessed: 2019-11-18.

[VBCG14] S. Varrette, P. Bouvry, H. Cartiaux, and F. Georgatos. Management of an academic HPC cluster: The UL experience. In *Proc. of the 2014 Intl. Conf. on High Performance Computing & Simulation (HPCS 2014)*, pages 959–967, Bologna, Italy, July 2014. IEEE.

[vDH00] Wim van Dam and Sean Hallgren. Efficient quantum algorithms for shifted quadratic character problems. *arXiv preprint quant-ph/0011067*, 2000.

[vS98] R. von Sterneck. Sur la distribution des résidus et des non-résidus quadratiques d'un nombre premier. *Matematicheskii Sbornik*, 20(2):269–284, 1898.

[vW94] Paul C. van Oorschot and Michael J. Wiener. Parallel collision search with application to hash functions and discrete logarithms. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, and Ravi S. Sandhu, editors, *ACM CCS 94*, pages 210–218. ACM Press, November 1994.

[Wag02] David Wagner. A generalized birthday problem. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 288–303. Springer, Heidelberg, August 2002.

[Wei48] André Weil. On some exponential sums. *Proceedings of the National Academy of Sciences of the United States of America*, 34(5):204, 1948.

Chapter 8

LegRoast and PorcRoast

Comme les quantités analogues à $N^{\frac{c-1}{2}}$ se rencontreront fréquemment dans le cours de nos recherches, nous emploierons le caractère abrégé $\left(\frac{N}{c}\right)$ pour exprimer le reste que donne $N^{\frac{c-1}{2}}$ divisé par c; reste qui, suivant ce qu'on vient de voir, ne peut être que $+1$ ou -1.

– Adrien-Marie Legendre, Essai sur la théorie des nombres, 1798

Publication Data

Beullens W., Delpech de Saint Guilhem C. LegRoast: Efficient Post-quantum Signatures from the Legendre PRF. In: Post-Quantum Cryptography. PQCrypto 2020. Lecture Notes in Computer Science, vol 12100, pages 130-150. Springer, 2020.

My Contribution

Main author. I designed the signature scheme, did the majority of the work writing the paper and the security proofs. I wrote the implementation and did the benchmarking.

LegRoast: Efficient post-quantum signatures from the Legendre PRF

Ward Beullens[1] and Cyprien Delpech de Saint Guilhem[1,2]

[1] imec-COSIC, KU Leuven, Belgium
[2] Dept Computer Science, University of Bristol, U.K.

Abstract. We introduce an efficient post-quantum signature scheme that relies on the one-wayness of the Legendre PRF. This “LEGendRe One-wAyness SignaTure” (LegRoast) builds upon the MPC-in-the-head technique to construct an efficient zero-knowledge proof, which is then turned into a signature scheme with the Fiat-Shamir transform. Unlike many other Fiat-Shamir signatures, the security of LegRoast can be proven without using the forking lemma, and this leads to a tight (classical) ROM proof. We also introduce a generalization that relies on the one-wayness of higher-power residue characters; the “POwer Residue ChaRacter One-wAyness SignaTure” (PorcRoast).

LegRoast outperforms existing MPC-in-the-head-based signatures (most notably Picnic/Picnic2) in terms of signature size and speed. Moreover, PorcRoast outperforms LegRoast by a factor of 2 in both signature size and signing time. For example, one of our parameter sets targeting NIST security level I results in a signature size of 7.2 KB and a signing time of 2.8ms. This makes PorcRoast the most efficient signature scheme based on symmetric primitives in terms of signature size and signing time.

Keywords: Post-Quantum signatures · Legendre PRF · MPC-in-the-head

This work was supported in part by the Research Council KU Leuven grants C14/18/067 and STG/17/019, by CyberSecurity Research Flanders with reference number VR20192203, by the ERC Advanced Grant ERC-2015-AdG-IMPaCT and by the Defense Advanced Research Projects Agency (DARPA) and Space and Naval Warfare Systems Center, Pacific (SSC Pacific) under contract No. N66001-15-C-4070. Ward Beullens is funded by an FWO fellowship.

1 Intoduction

In 1994, Shor discovered a quantum algorithm for factoring integers and solving discrete logarithms in polynomial time [26]. This implies that an adversary with access to a sufficiently powerful quantum computer can break nearly all public-key cryptography that is deployed today. Therefore, it is important to look for alternative public-key cryptography algorithms that can resist attacks from quantum adversaries. Recently, the US National Institute of Standards and Technology (NIST) has initiated a process to solicit, evaluate, and standardize one or more quantum-resistant public-key cryptographic algorithms [22]. One of the 9 signature schemes that advanced to the second round of the NIST project is Picnic [7,19,27], a signature scheme whose security only relies on symmetric-key primitives.

Indeed, a key pair for Picnic consists of a random secret key sk and the corresponding public key $\mathsf{pk} = F(\mathsf{sk})$, where F is a one-way function which can be computed with a low number of non-linear binary gates [7]. To sign a message m the signer then produces a non-interactive zero-knowledge proof of knowledge of sk such that $F(\mathsf{sk}) = \mathsf{pk}$ in a way that binds the message m to the proof. These zero-knowledge proofs (whose security relies additionally only on a secure commitment scheme) are constructed using the MPC-in-the-head paradigm [17]. This results in a signature scheme whose signatures are 33 KB large for 128 bits of security. Later, Katz et al. developed Picnic2 [19], which reduces the signature size to only 14 KB by moving from a 3-party MPC protocol in the honest majority setting to an n-party protocol with preprocessing secure in the dishonest majority setting. However, this increased number of parties slows down the signing and verification algorithms. Picnic and Picnic2 are round 2 candidates in the NIST project [27]. To study the effect of selecting a different function F, Delpech de Saint Guilhem et al. constructed the BBQ scheme using MPC protocols for arithmetic secret sharing to base the signatures on the security of the AES algorithm instead of the less scrutinized block cipher LowMC [24].

Contributions. In this work we propose to use the Legendre PRF [9], denoted by $\mathcal{L}_K(\cdot)$, as one-way function, instead of LowMC or AES. The Legendre PRF is a promising alternative since it can be computed very efficiently in the MPC setting [15]. However, a major limitation of the Legendre PRF is that it only produces one bit of output, which means that the public key should consist of many PRF evaluations $\mathcal{L}_K(i_1), \dots, \mathcal{L}_K(i_L)$, at some fixed arbitrary list $\mathcal{I} = (i_1, \cdots, i_L)$ of L elements of $\mathbb{F}_p$, to uniquely determine the secret key K. Hence, the zero-knowledge proof needs to prove knowledge of a value K' such that $\mathcal{L}_{K'}(i) = \mathcal{L}_K(i)$ for all $i \in \mathcal{I}$ simultaneously, which results in prohibitively large signatures. Luckily, we can relax the relation to overcome this problem. Instead of proving that the signer knows a K' such that $\mathcal{L}_{K'}(i) = \mathcal{L}_K(i)$ for all $i \in \mathcal{I}$, we let a prover prove knowledge of a K' such that this holds for a large fraction of the i in $\mathcal{I}$. We show that the relaxed statement allows for a much more efficient zero-knowledge proof. This allows us to establish LegRoast, an MPC-in-the-head based scheme with a signature size of 12.2 KB and with much faster signing and verification algorithms than the Picnic2 and BBQ schemes. To further improve the efficiency of LegRoast, we propose to use higher-power residuosity symbols instead of just the quadratic one (i.e. the Legendre symbol) in a second scheme called PorcRoast. This results in signatures that are only 6.3 KB large and in signing and verification times that are twice faster than LegRoast.

A comparison between the signature size and signing time of LegRoast and PorcRoast versus existing signatures based on symmetric primitives (Picnic [27] and SPHINCS+ [16]) is shown in Figure 1. Even though LegRoast and PorcRoast do not have an AVX optimized implementation yet, we see that LegRoast has faster signing times than both Picnic and SPHINCS+, and that PorcRoast is even faster than LegRoast. We conclude that PorcRoast is the most efficient post-quantum signature scheme based on symmetric primitives in terms of signature size and signing time.

However, note that there are several other branches of post-quantum signatures, such as lattice-based (e.g. Dilithium and Falcon [12,21,23]), Multivariate signatures (e.g., Rainbow, LUOV, MQDSS, MUDFISH [11,10,5,6,25,2]) and isogeny-based signatures (e.g. CSI-FISH [4]), some of which result in more efficient signature schemes.

Roadmap. After some preliminaries in Section 2, we introduce a relaxed PRF relation in Section 3. We then sketch an identification scheme in Section 4 which we formalize as a signature scheme in Section 5. We finally discuss parameter choices and implementation results in Section 6.

2 Preliminaries - the Legendre and power residue PRFs

For an odd prime p the Legendre PRF is conjectured to be a pseudorandom function family, indexed by a key $K \in \mathbb{Z}_p$, such that $\mathcal{L}_K$ takes as input an element $a \in \mathbb{F}_p$ and outputs the bit

$$\mathcal{L}_K(a) = \left\lfloor \frac{1}{2}\left(1 - \left(\frac{K+a}{p}\right)\right) \right\rceil \in \mathbb{Z}_2,$$

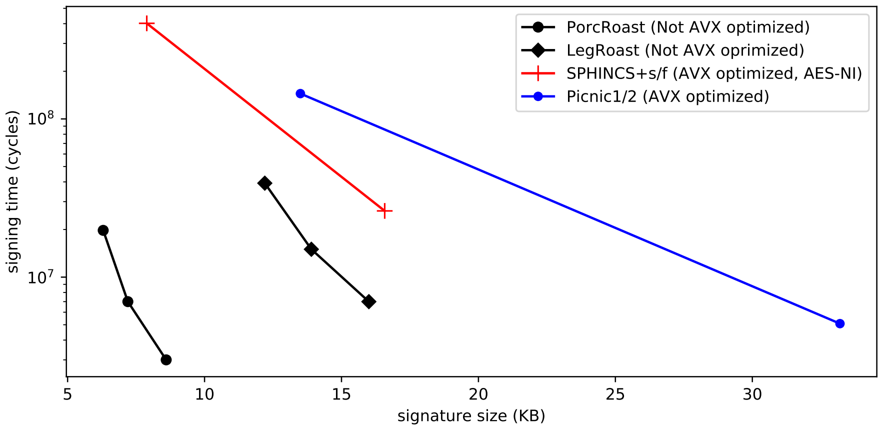


Fig. 1. Signature sizes and timings of post-quantum signature schemes based only on symmetric primitives.

where $(\frac{a}{p}) \in \{-1, 0, 1\}$ denotes the quadratic residuosity symbol of a mod p. We note that the function $\mathcal{L}_K$ above is defined such that $\mathcal{L}_0(a \cdot b) = \mathcal{L}_0(a) + \mathcal{L}_0(b)$ for all $a, b \in \mathbb{F}_p^\times$. (Note also that $\mathcal{L}_K(a) = \mathcal{L}_0(K + a)$.)

The seemingly random properties of quadratic residues have been the subject of study for number theorists at least since the early twentieth century, which is why Damgård proposed to use this construction in cryptography [9]. Since then, the security of the Legendre PRF has been studied in several attack models. In the very strong model where a quantum adversary is allowed to query the PRF in superposition, a key can be recovered in quantum polynomial time [8]. If the adversary is only allowed to query the PRF classically, there is a memoryless classical attack that requires computing $O(p^{1/2} \log p)$ Legendre symbols and making $O(p^{1/2} \log p)$ queries to the PRF [20]. Finally, if the adversary is restricted to querying only L Legendre symbols, the best known attack requires computing $O(p \log^2 p / L^2)$ Legendre symbols [3].

Damgård also considers a generalisation of the Legendre PRF, where instead of using the quadratic residue symbol $(\frac{a}{p}) = a^{\frac{p-1}{2}} \bmod p$, the PRF uses the k-th power residue symbol defined as $(\frac{a}{p})_k = a^{\frac{p-1}{k}} \bmod p$, for some k that divides $p - 1$. We define the power residue PRF, analogous to the Legendre PRF, as the keyed function $\mathcal{L}_K^k : \mathbb{F}_p \to \mathbb{Z}_k$, where for an odd prime $p \equiv 1 \bmod k$, $\mathcal{L}_K^k(a)$

is defined as

$$\mathcal{L}_K^k(a) = \begin{cases} i & \text{if } (a+K)/g^i \equiv h^k \bmod p \text{ for some } h \in \mathbb{F}_p^\times \\ 0 & \text{if } (a+K) \equiv 0 \bmod p \end{cases},$$

where g is a fixed generator of $\mathbb{F}_p^\times$. We see that the function $\mathcal{L}_0^k$ is a homomorphism of groups from $\mathbb{F}_p^\times$ to $\mathbb{Z}_k$.

Note that for $k = 2$, this notation coincides with the original Legendre PRF. In this paper, we use the generic notation and we separate the $k = 2$ and $k > 2$ cases only in the experimental sections to highlight the advantages gained by using $k > 2$. One advantage of the power residue PRF is that it yields $\log k$ bits of output, instead of a single bit. The best known attack against the power residue PRF in the setting where an attacker is allowed to query the PRF L times requires computing $O(p \log^2 p/(kL \log^2 k))$ power residue symbols [3].

3 The (relaxed) power residue PRF relation

In this section, we define NP-languages $R_{\mathcal{L}^k}$ for the Legendre PRF ($k = 2$) and higher power residue PRF ($k > 2$), which consist of the symbol strings of outputs of the $\mathcal{L}^k$ PRF for a given set of inputs. We also define a relaxed version of these languages $R_{\beta\mathcal{L}^k}$, which consist of the strings that are very close (up to addition by a scalar in $\mathbb{Z}_k$) to a word in $R_{\mathcal{L}^k}$, where the Hamming distance d_H is used and β parameterizes the slack.

For properly chosen parameters, it follows from the Weil bound that the relaxed version is as hard as the exact relation, but the relaxed relation will lead to much more efficient signature schemes. To simplify notation, for a list $\mathcal{I} = (i_1, \cdots, i_L)$ of L arbitrary elements of $\mathbb{Z}_p$, we denote a length-L Legendre / k-th power residue PRF as:

$$\begin{aligned} F_{\mathcal{I}}^k : \mathbb{F}_p &\to \mathbb{Z}_k^L \\ K &\mapsto (\mathcal{L}_K^k(i_1), \dots, \mathcal{L}_K^k(i_L)). \end{aligned}$$

Definition 1 (Legendre / k-th power residue PRF relation). *For an odd prime p, a positive integer $k \mid p-1$ and a list $\mathcal{I}$ of L elements of $\mathbb{Z}_p$ we define the Legendre / k-th power residue PRF relation $R_{\mathcal{L}^k}$ with output length L as*

$$R_{\mathcal{L}^k} = \left\{ (F_{\mathcal{I}}^k(K), K) \in \mathbb{Z}_k^L \times \mathbb{F}_p \mid K \in \mathbb{F}_p \right\}.$$

Definition 2 (β-approximate PRF relation). *For $\beta \in [0,1]$, an odd prime p, a positive integer $k \mid p-1$ and a list $\mathcal{I}$ of L elements of $\mathbb{Z}_p$ we define the β-approximate PRF relation $R_{\beta\mathcal{L}^k}$ with output length L as*

$$R_{\beta\mathcal{L}^k} = \{(s,K) \in \mathbb{Z}_k^L \times \mathbb{F}_p \mid \exists a \in \mathbb{Z}_k : d_H(s + (a,\ldots,a), F_{\mathcal{I}}^k(K)) \leq \beta L\}$$

where $d_H(\cdot,\cdot)$ denotes the Hamming distance.

It follows from the Weil bound for character sums that if β is sufficiently small and L is sufficiently large, then the β-relaxed power residue relation is equally hard as the exact power residue relation, simply because with overwhelming probability over the choice of $\mathcal{I} = (i_1, \cdots, i_L)$ every witness for the relaxed relation is also a witness for the exact relation. The proof is given in Appendix A.

Theorem 1. *Let $\mathcal{B}(n,q)$ denote the binomial distribution with n samples each with success probability q. Take $K \in \mathbb{F}_p$, and take $s = F_{\mathcal{I}}^k(K)$. Then with probability at least $1 - kp \cdot \Pr\left[\mathcal{B}(L, 1/k + 1/\sqrt{p} + 2/p) \geq (1-\beta)L\right]$ over the choice of $\mathcal{I}$, there exist only one witness for $s \in R_{\beta\mathcal{L}^k}$, namely K, which is also a witness for the exact relation $R_{\mathcal{L}^k}$.*

4 Identification scheme

In this section, we establish a Picnic-style identification scheme from the Legendre / k-th power residue PRF. We first sketch a scheme very close to the original Picnic construction [7] and gradually add more optimizations, presenting each in turn. Even though the final goal is to construct a signature scheme, we use the language of identification schemes in this section to relate the scheme to existing constructions. We delay the security proof to the next section, where we first apply the Fiat-Shamir transform [13] before we prove that the resulting signature scheme is tightly secure in the ROM. The proof of security of the interactive identification scheme presented here can be derived from the one provided in the next section.

Starting point. To begin, we take the Picnic2 identification scheme and replace the LowMC block-cipher by the PRF $F_{\mathcal{I}}^k$. The key pair is then $\mathsf{sk} = K$ and $\mathsf{pk} = F_{\mathcal{I}}^k(K) \in \mathbb{Z}_k^L$. From a high-level view, the protocol can be sketched as in Figure 2 where the prover runs an MPC-in-the-head proof with N parties on a secret sharing of K, to prove to the verifier that he knows K such that

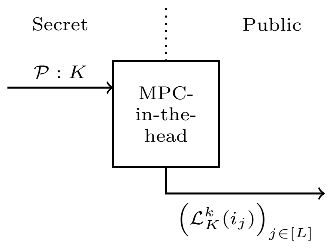
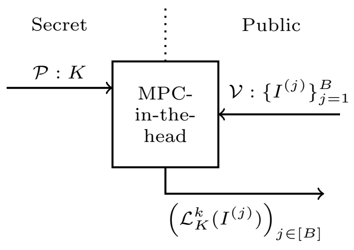


Fig. 2. Picnic-stye identification scheme



Secret
Public
$\mathcal{P} : K$
MPC-in-the-head
$\mathcal{V} : \{I^{(j)}\}_{j=1}^B$
$\left(\mathcal{L}^k_K(I^{(j)})\right)_{j\in[B]}$

Fig. 3. Checking only B symbols

$((\frac{K+i_1}{p}),\ldots,(\frac{K+i_L}{p}))$ is equal to the public key. We also use the more efficient method recently proposed by Baum and Nof [1] based on sacrificing rather than the cut-and-choose technique.

Relaxing the PRF relation. As a first optimization, rather than computing all of the L residue symbols with the MPC protocol, we only check a fixed number B of them. To do so, the verifier chooses random inputs $I^{(1)},\ldots,I^{(B)}$ in $\mathcal{I}$ at which the $\mathcal{L}^k$ PRF is evaluated to check the witness. It is crucial that the verifier sends his choice of $I^{(j)}$s after the prover has committed to his sharing of K, because if a malicious prover knows beforehand which symbols are going to be checked, he can use a fake key K' such that $(\frac{K'+I^{(j)}}{p}) = \mathsf{pk}_{I^{(j)}}$ only for $j \in [B]$. This probabilistic method of selecting which circuit will be executed with the MPC-in-the-head technique is similar to the "sampling circuits on the fly" technique of Baum and Nof [1].

This is now an identification scheme for the β-approximate Legendre PRF relation; a prover that convinces the verifier with probability greater than $(1-\beta)^B + (1-(1-\beta)^B)/N$ could be used to extract a β-approximate witness following the formalism presented in [1, Section 4]. This protocol is sketched in Figure 3.

Computing residue symbols in the clear. Since computing residue symbols is relatively expensive, we avoid doing it within the MPC protocol. We use an idea similar to that of Grassi et al. to make this possible [15]. First, we let the prover create sharings of B uniformly random values $r^{(1)},\ldots,r^{(B)} \in \mathbb{F}_p^\times$ and commit to their residue symbols by sending $s^{(j)} = \mathcal{L}^k_0(r^{(j)})$ to the verifier. Then, the MPC protocol only outputs $o^{(j)} = (K + I^{(j)})r^{(j)}$. Since $K + I^{(j)}$ is masked with a uniformly random value with known residue symbol, $o^{(j)}$ does not leak information about K (except for the residue symbol of $K + I^{(j)}$). The verifier then computes $\mathcal{L}_0^k(o^{(j)})$ himself in the clear, and verifies whether it equals $\mathsf{pk}_{I^{(j)}} + s^{(j)}$. The correctness of this check follows from the facts that $\mathcal{L}_0^k : \mathbb{F}_p^\times \to \mathbb{Z}_k$ is a group homomorphism.

Note that the prover can lie about the values of $s^{(j)} = \mathcal{L}_0^k(r^{(j)})$ that he sends to the prover. This is not an issue because he has to commit to these values before the choice of $I^{(j)}$ is revealed. This is the reason why we defined K' to be an β-approximate witness for pk if $F_{\mathcal{I}}^k(K')$ is close to $\mathsf{pk} = F_{\mathcal{I}}^k(K)$ *up to addition by a scalar*. This identification protocol is sketched in Figure 4.

Verifying instead of computing multiplications. Instead of using the MPC protocol to *compute* the products $o^{(j)}$, the prover can just send these products directly to verifier. We then use the MPC-in-the-head protocol to instead *verify* that $o^{(j)} = (K + I^{(j)}) \cdot r^{(j)}$ for all $j \in [B]$. A big optimization here is that rather than verifying these B equations separately, it is possible to just check a random linear combination of these equations:

After the prover sends the $o^{(j)}$ values, the verifier chooses random coefficients $\lambda^{(1)}, \dots, \lambda^{(B)}$ for the linear combination. Then, the MPC protocol is used to compute the error term E defined as

$$E = \sum_{j=1}^{B} \lambda^{(j)} \left((K + I^{(j)}) r^{(j)} - o^{(j)} \right) = K \cdot \sum_{j=1}^{B} \lambda^{(j)} r^{(j)} + \sum_{j=1}^{B} \lambda^{(j)} (I^{(j)} r^{(j)} - o^{(j)}).$$

Clearly, if all the $o^{(j)}$ are correct, then $E = 0$. Otherwise, if one or more of the $o^{(j)}$ are wrong, then E will be a uniformly random value. Therefore, checking if $E = 0$ proves to the verifier that all the $o^{(j)}$ are correct, with a soundness error of $1/p$. Moreover, since the $\lambda^{(j)}, o^{(j)}$ and $I^{(j)}$ are public values, we see that E can be computed with only a single nonlinear operation! This means we can compute E extremely efficiently in MPC. The identification scheme with this final optimization is sketched in Figure 5.

We note that a single execution of the interactive identification scheme is not enough to achieve negligible soundness error (e.g. the prover has probability $1/N$ to cheat in the MPC verification protocol). To resolve this, M executions must be run in parallel.

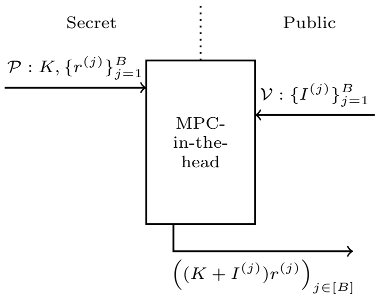

Secret
Public
$\mathcal{P} : K, \{r^{(j)}\}_{j=1}^{B}$
$\mathcal{V} : \{I^{(j)}\}_{j=1}^{B}$
MPC-in-the-head
$\left((K + I^{(j)})r^{(j)}\right)_{j \in [B]}$

Fig. 4. Computations in the clear.

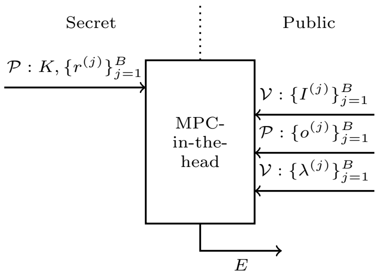

Secret
Public
$\mathcal{P} : K, \{r^{(j)}\}_{j=1}^{B}$
$\mathcal{V} : \{I^{(j)}\}_{j=1}^{B}$
$\mathcal{P} : \{o^{(j)}\}_{j=1}^{B}$
$\mathcal{V} : \{\lambda^{(j)}\}_{j=1}^{B}$
MPC-in-the-head
E

Fig. 5. The final scheme.

5 LegRoast and PorcRoast signature schemes

We now formalize the signature schemes LegRoast (with $k = 2$) and PorcRoast (with $k > 2$) which are constructed from the identification scheme of Section 4 with the Fiat-Shamir transform [13], by generating the challenges using three random oracles $\mathcal{H}_1, \mathcal{H}_2$ and $\mathcal{H}_3$. The message is combined with a 2λ-bit salt and bound to the proof by hashing it together with the messages of the prover.

Parameters. Our new signature schemes are parametrized by the following values. Let p be a prime number and let $k \geq 2$ be an integer such that $k \mid p-1$. Let L be an integer determining the length of the public key, $\mathcal{I}$ a pseudo-randomly chosen list of L elements of $\mathbb{Z}_p$ and let $B \leq L$ denote the number of k-th power residue symbols in the public key that will be checked at random. Let N denote the number of parties in the MPC verification protocol and let M denote the number of parallel executions of the identification scheme. These values are grouped under the term params.

Key generation, signing and verifying. The $\mathsf{KGen}(1^\lambda, \mathsf{params})$ algorithm samples $\mathsf{sk} = K \xleftarrow{\$} \mathbb{F}_p$ uniformly at random and computes the public key $\mathsf{pk} = F_{\mathcal{I}}^{k}(K)$. The $\mathsf{Sign}(\mathsf{params}, \mathsf{sk}, m)$ algorithm, for message $m \in \{0,1\}^*$ is presented in Figure 6. The $\mathsf{Vf}(\mathsf{params}, \mathsf{pk}, m, \sigma)$ algorithm is presented in Figure 7.

Security. The EUF-CMA security [14] of the LegRoast and PorcRoast signature schemes follows from a *tight* reduction from the problem of finding a witness for the $R_{\beta\mathcal{L}^k}$-relation, which is equally hard as a key recovery on the power residue PRF for our parameters. The proof of Theorem 2 is included in Appendix B.

Theorem 2. *In the classical random oracle model, the LegRoast and PorcRoast signature schemes defined as above are EUF-CMA-secure under the assumption that computing β-approximate witnesses for a given public key is hard.*

6 Parameter choices and implementation

This section shows how to choose secure parameters for the LegRoast and PorcRoast signature schemes, and what the resulting key and signature sizes are. We also go over some of the implementation details and the performance of our implementation.

6.1 Parameter choices

Choosing p, L and $\mathcal{I}$. We choose p and L such that the problem of finding a β-approximate witness for the PRF relation has the required security level. To do this, we first choose p and L such that the problem of recovering the exact key from L symbols of output is hard. For our proposed parameters we choose L such that the public key size is 4KB, (i.e. $L = 32768/\log(k)$). Different trade-offs are possible (see remark 1). Then, we set β such that

$$k \cdot p \cdot \Pr[B(L, 1/k + 1/\sqrt(p) + 2/p) > (1-\beta)l] \leq 2^{-\lambda}\,.$$

With this choice, Theorem 1 says that with overwhelming probability, finding a β-approximate key is equivalent to finding the exact key. Section 2 gives a short overview of attacks on the Legendre PRF for various attack models. However, in the setting of attacking LegRoast and PorcRoast, the adversary is restricted even more than in the weakest attacker model considered in the literature: an attacker learns only a few evaluations of the Legendre PRF on pseudorandom inputs over which the attacker has no control. If the L inputs are chosen at random, the best known attack is a brute force search which requires computing $O(p/k)$ power residue symbols, and the attack complexity becomes independent of L. For Legroast, we propose to use a prime p of size roughly 2^{λ}, where λ is the required security level. We choose the Mersenne prime $p = 2^{127} - 1$ to speed up the arithmetic. For PorcRoast, we use the same prime and $k = 254$ such that a power residue symbol can efficiently be represented by a single byte. For $k > 2$, computing a power residue symbol corresponds to a modular exponentiation, which is much more expensive than an AES operation, so even

```
$\mathsf{Sign}(\mathsf{params}, \mathsf{sk}, m)$ :
**Phase 1: Commitment to sharings of $K$, randomness and triples**
 1: Pick a random salt: $\mathsf{salt} \leftarrow \{0,1\}^{2\lambda}$.
 2: **for** $e$ from 1 to $M$ **do**
 3:     Sample a root seed: $\mathsf{sd}_e \xleftarrow{\$} \{0,1\}^{\lambda}$.
 4:     Build binary tree from $\mathsf{sd}_e$ with leaves $\mathsf{sd}_{e,1}, \dots, \mathsf{sd}_{e,N}$.
 5:     **for** $i$ from 1 to $N$ **do**
 6:         Sample shares: $K_{e,i}, r_{e,i}^{(1)}, \dots, r_{e,i}^{(B)}, a_{e,i}, b_{e,i}, c_{e,i} \leftarrow \mathsf{Expand}(\mathsf{sd}_{e,i})$.
 7:         Commit to seed: $\mathsf{C}_{e,i} \leftarrow \mathcal{H}_{\mathsf{sd}}(\mathsf{salt}, e, i, \mathsf{sd}_{e,i})$.
 8:     Compute witness offset: $\Delta K_e \leftarrow K - \sum_{i=1}^{N} K_{e,i}$.
 9:     Adjust first share: $K_{e,1} \leftarrow K_{e,1} + \Delta K_e$.
10:     Compute triple: $a_e \leftarrow \sum_{i=1}^{N} a_{e,i}$, $b_e \leftarrow \sum_{i=1}^{N} b_{e,i}$ and $c_e \leftarrow a_e \cdot b_e$.
11:     Compute triple offset: $\Delta c_e \leftarrow c_e - \sum_{i=1}^{N} c_{e,i}$.
12:     Adjust first share: $c_{e,1} \leftarrow c_{e,1} + \Delta c_e$.
13:     **for** $j$ from 1 to $B$ **do**
14:         Compute residuosity symbol: $s_e^{(j)} \leftarrow \mathcal{L}_0^k(r_e^{(j)})$ where $r_e^{(j)} \leftarrow \sum_{i=1}^{N} r_{e,i}^{(j)}$.
15: Set $\sigma_1 \leftarrow ((\mathsf{C}_{e,i})_{i\in[N]}, (s_e^{(j)})_{j\in[B]}, \Delta K_e, \Delta c_e)_{e\in[M]}$.
**Phase 2: Challenge on public key symbols**
 1: Compute challenge hash: $h_1 \leftarrow \mathcal{H}_1(m, \mathsf{salt}, \sigma_1)$.
 2: Expand hash: $(I_e^{(j)})_{e\in[M],j\in[B]} \leftarrow \mathsf{Expand}(h_1)$, where $I_e^{(j)} \in \mathcal{I}$.
**Phase 3: Computation of output values**
 1: **for** $e$ from 1 to $M$ and **for** $j$ from 1 to $B$ **do**
 2:     Compute output value: $o_e^{(j)} \leftarrow (K + I_e^{(j)}) \cdot r_e^{(j)}$.
 3: Set $\sigma_2 \leftarrow (o_e^{(1)}, \dots, o_e^{(B)})_{e\in[M]}$.
**Phase 4: Challenge for sacrificing-based verification**
 1: Compute challenge hash: $h_2 \leftarrow \mathcal{H}_2(h_1, \sigma_2)$.
 2: Expand hash $(\epsilon_e, \lambda_e^{(1)}, \dots, \lambda_e^{(B)})_{e\in[M]} \leftarrow \mathsf{Expand}(h_2)$, where $\epsilon_e, \lambda_e^{(j)} \in \mathbb{Z}_p$.
**Phase 5: Commitment to views of sacrificing protocol**
 1: **for** $e$ from 1 to $M$ **do**
 2:     **for** $i$ from 1 to $N$ **do**
 3:         Compute shares: $\alpha_{e,i} \leftarrow a_{e,i} + \epsilon_e K_{e,i}$ and $\beta_{e,i} \leftarrow b_{e,i} + \sum_{j=1}^{B} \lambda_e^{(j)} r_{e,i}^{(j)}$.
 4:     Compute values: $\alpha_e \leftarrow \sum_{i=1}^{N} \alpha_{e,i}$ and $\beta_e \leftarrow \sum_{i=1}^{N} \beta_{e,i}$.
 5:     **for** $i$ from 1 to $N$ **do**
 6:         Compute product shares: $z_{e,i} \leftarrow \sum_{j=1}^{B} -\lambda_e^{(j)} r_{e,i}^{(j)} I_e^{(j)}$.
 7:         **if** $i \overset{?}{=} 1$ **then** $z_{e,i} \leftarrow z_{e,i} + \sum_{j=1}^{B} \lambda_e^{(j)} o_e^{(j)}$.
 8:         Compute check value shares: $\gamma_{e,i} \leftarrow \alpha_e b_{e,i} + \beta_e a_{e,i} - c_{e,i} + \epsilon_e z_{e,i}$.
 9: Set $\sigma_3 \leftarrow (\alpha_e, \beta_e, (\alpha_{e,i}, \beta_{e,i}, \gamma_{e,i})_{i\in[N]})_{e\in[M]}$.
**Phase 6: Challenge on sacrificing protocol**
 1: Compute challenge hash $h_3 \leftarrow \mathcal{H}_3(h_2, \sigma_3)$.
 2: Expand hash $(\bar{i}_e)_{e\in[M]} \leftarrow \mathsf{Expand}(h_3)$, where $\bar{i}_e \in [N]$.
**Phase 7: Opening the views of sacrificing protocol**
 1: **for** $e$ from 1 to $M$ **do**
 2:     $\mathsf{seeds}_e \leftarrow \{\log_2(N)$ nodes in tree needed to compute $\mathsf{sd}_{e,i}$ for $i \in [N] \setminus \bar{i}\}$
```

```
Vf(params, pk, $m, \sigma$):
 1: Parse $\sigma = (\mathsf{salt}, h_1, h_3, (\Delta K_e, \Delta c_e, o_e^{(1)}, \ldots, o_e^{(B)}, \alpha_e, \beta_e, \mathsf{seeds}_e, \mathsf{C}_{e,\bar{i}_e})_{e\in[M]})$.
 2: Compute $h_2 \leftarrow \mathcal{H}_2(h_1, (o_e^{(j)})_{e\in[M],j\in[B]})$.
 3: Expand challenge hash 1: $(I_e^{(1)}, \ldots, I_e^{(B)})_{e\in[M]} \leftarrow \mathsf{Expand}(h_1)$, where $I_e^{(j)} \in \mathcal{I}$.
 4: Expand challenge hash 2: $(\epsilon_e, \lambda_e^{(1)}, \ldots, \lambda_e^{(B)})_{e\in[M]} \leftarrow \mathsf{Expand}(h_2)$.
 5: Expand challenge hash 3: $(\bar{i}_e)_{e\in[M]} \leftarrow \mathsf{Expand}(h_3)$.
 6: for $e$ from 1 to $M$ do
 7:     Use $\mathsf{seeds}_e$ to compute $\mathsf{sd}_{e,i}$ for $i \in [N] \setminus \bar{i}_e$.
 8:     for $i$ from 1 to $\bar{i}_e - 1$ and from $\bar{i}_e + 1$ to $N$ do
 9:         Sample shares: $K_{e,i}, r_{e,i}^{(1)}, \ldots, r_{e,i}^{(B)}, a_{e,i}, b_{e,i}, c_{e,i} \leftarrow \mathsf{Expand}(\mathsf{sd}_{e,i})$.
10:         if $i \overset{?}{=} 1$ then
11:             Adjust shares: $K_{e,i} \leftarrow K_{e,i} + \Delta K_e$ and $c_{e,i} \leftarrow c_{e,i} + \Delta c_e$.
12:         Recompute commitments: $\mathsf{C}_{e,i}^* \leftarrow \mathcal{H}(\mathsf{salt}, e, i, \mathsf{sd}_{e,i})$
13:         Recompute shares: $\alpha_{e,i}^* \leftarrow a_{e,i} + \epsilon_e K_{e,i}$ and $\beta_{e,i}^* \leftarrow b_{e,i} + \sum_{j=1}^{B} \lambda_e^{(j)} r_{e,i}^{(j)}$.
14:         Recompute product shares: $z_{e,i} \leftarrow \sum_{j=1}^{B} -\lambda_e^{(j)} r_{e,i}^{(j)} I_e^{(j)}$.
15:         if $i \overset{?}{=} 1$ then
16:             $z_{e,i} \leftarrow z_{e,i} + \sum_{j=1}^{B} \lambda_e^{(j)} o_e^{(j)}$.
17:         Recompute check value shares: $\gamma_{e,i}^* \leftarrow \alpha_e b_{e,i} + \beta_e a_{e,i} - c_{e,i} + \epsilon_e z_{e,i}$.
18:     Compute missing shares: $\alpha_{e,\bar{i}_e}^* \leftarrow \alpha_e - \sum_{i\neq\bar{i}} \alpha_{e,i}^*$ and $\beta_{e,\bar{i}_e}^* \leftarrow \beta_e - \sum_{i\neq\bar{i}} \beta_{e,i}^*$.
19:     Compute missing check value share: $\gamma_{e,\bar{i}_e}^* = \alpha_e\beta_e - \sum_{i\neq\bar{i}} \gamma_{e,i}^*$.
20:     for $j$ from 1 to $B$ do
21:         Recompute residuosity symbols: $s_e^{(j)*} \leftarrow \mathcal{L}_0^k(o_e^{(j)}) - \mathsf{pk}_{I_e^{(j)}}$.
22: Check 1: $h_1 \overset{?}{=} \mathcal{H}_1(m, \mathsf{salt}, ((\mathsf{C}_{e,i}^*)_{i\in[N]}, (s_e^{(j)*})_{j\in[B]}, \Delta K_e, \Delta c_e)_{e\in[M]})$
23: Check 2: $h_3 \overset{?}{=} \mathcal{H}_3(h_2, (\alpha_e, \beta_e, (\alpha_{e,i}^*, \beta_{e,i}^*, \gamma_{e,i}^*)_{i\in[N]})_{e\in[M]})$
24: Output accept if both checks pass.
```

Fig. 7. Verifying algorithm for LegRoast and PorcRoast.

though an attacker has on average only to compute $2^{127}/k \approx 2^{119}$ power residue symbols, we claim that this still provides approximately 128-bits of security. We stress that the quantum polynomial-time key recovery attack on the Legendre PRF does not apply on our scheme, because the adversary can not make queries to the instance of the Legendre PRF (and certainly no quantum queries) [8].

Choosing *B*, *N* and *M*. Our security proof shows that, unless an attacker can produce a β-approximate witness, his best strategy is to query $\mathcal{H}_1$ on many inputs and then choose the query for which

$$\mathcal{L}_0^k((K_e + I_e^{(j)})r_e^{(j)}) = s_e^{(j)} + \mathsf{pk}_{I_e^{(j)}} \text{ for all } j \in [B]$$

holds for the most executions. Say this is the case for M' out of M executions. He then makes one of the parties cheat in the MPC protocol in each of the $M - M'$ remaining executions and queries $\mathcal{H}_3$ in the hope of getting an output $\{\bar{i}_e\}_{e\in[M]}$ that asks him to open all the other non-cheating parties; i.e. the attacker attempts to guess $\bar{i}_e$ for each e. This succeeds with probability $N^{-M+M'}$.

Therefore, to achive λ bits of security, we take parameters $B, N = 2^n$ and M such that

$$\min_{M'\in\{0,\dots,M\}} \left(\Pr[\mathcal{B}(M,(1-\beta)^B) \geq M_1]^{-1} + N^{M-M'}\right) \geq 2^\lambda\,, \tag{1}$$

which says that for each value of M', the adversary is expected to do at least 2^λ hash function evalutations for the attack to succeed. To choose parameters, we fix N to a certain value and compute which values of B and M minimize the signature size while satisfying Equation (1). The choice of N controls a trade-off between signing time and signature size. If N is large, the soundness error will be small, which results in a smaller signature size, but the signer and the verifier need to simulate an MPC protocol with a large number of parties, which is slow. On the other hand, if N is small, then the signature size will be larger, but signing and verifying will be faster. Some trade-offs achieving 128-bits of security for LegRoast and PorcRoast are displayed in Table 1.

Remark 1. The parameter L controls a trade-off between public key size and signature size. For example, we can decrease the public key size by a factor 8 (to 0.5KB), at the cost of an increase in signature size by 21% (to 7.6 KB). ($L = 512, k = 254, \beta = 0.871, n = 256, B = 10, M = 20$).

	Parameters			Signature Size	Signing time
	N	M	B	(KB)	(ms)
LegRoast	16	54	9	16.0	2.8
$k = 2$	64	37	12	13.9	6.0
$\beta = 0.449$	256	26	16	12.2	15.7
PorcRoast	16	39	4	8.6	1.2
$k = 254$	64	27	5	7.2	2.8
$\beta = 0.967$	256	19	6	6.3	7.9

Table 1. Parameter sets for LegRoast and PorcRoast for NIST security level I. For all parameter sets we have $p = 2^{127} - 1$, a secret key size of 16 Bytes and a public key size of 4 KB ($L = 32768$ and 4096 for LegRoast and PorcRoast respectively). The verification time is similar to the signing time.

6.2 Implementation

In our implementation, which is publicly available at

`https://github.com/WardBeullens/LegRoast`.

we replace the random oracles and the Expand function by the SHA-3 and SHAKE128 hash functions. The signing algorithm can easily be implemented in a constant time manner, except for computing Legendre symbols, which when implemented with the usual GCD strategy, leaks timing information on its argument. Therefore, in our implementation, we chose to adopt the slower approach of computing Legendre symbols as an exponentiation with fixed exponent $(p-1)/2$, which is can be implemented in constant time more easily. Higher-power residue symbols are also calculated as an exponentiation with fixed exponent $(p-1)/k$. The signing-time of our implementation, measured on an Intel i5-8400H CPU, running at 2.50GHz, is displayed in Table 1.

References

1. Baum, C., Nof, A.: Concretely-efficient zero-knowledge arguments for arithmetic circuits and their application to lattice-based cryptography. Cryptology ePrint Archive, Report 2019/532 (2019), `https://eprint.iacr.org/2019/532`
2. Beullens, W.: Sigma protocols for mq, pkp and sis, and fishy signature schemes. Cryptology ePrint Archive, Report 2019/490 (2019), `https://eprint.iacr.org/2019/490`
3. Beullens, W., Beyne, T., Udovenko, A., Vitto, G.: Cryptanalysis of the legendre prf and generalizations. Cryptology ePrint Archive, Report 2019/1357 (2019), `https://eprint.iacr.org/2019/1357`
4. Beullens, W., Kleinjung, T., Vercauteren, F.: Csi-fish: Efficient isogeny based signatures through class group computations. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 227–247. Springer (2019)
5. Beullens, W., Preneel, B.: Field lifting for smaller UOV public keys. In: Patra, A., Smart, N.P. (eds.) INDOCRYPT 2017. LNCS, vol. 10698, pp. 227–246. Springer, Heidelberg (Dec 2017)
6. Beullens, W., Preneel, B., Szepieniec, A., Vercauteren, F.: LUOV. Tech. rep., National Institute of Standards and Technology (2019), available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions`
7. Chase, M., Derler, D., Goldfeder, S., Orlandi, C., Ramacher, S., Rechberger, C., Slamanig, D., Zaverucha, G.: Post-quantum zero-knowledge and signatures from symmetric-key primitives. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) ACM CCS 2017. pp. 1825–1842. ACM Press (Oct / Nov 2017)

8. van Dam, W., Hallgren, S.: Efficient quantum algorithms for shifted quadratic character problems. arXiv preprint quant-ph/0011067 (2000)
9. Damgård, I.: On the randomness of legendre and jacobi sequences. In: Goldwasser, S. (ed.) CRYPTO'88. LNCS, vol. 403, pp. 163–172. Springer, Heidelberg (Aug 1990)
10. Ding, J., Chen, M.S., Petzoldt, A., Schmidt, D., Yang, B.Y.: Rainbow. Tech. rep., National Institute of Standards and Technology (2019), available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions`
11. Ding, J., Schmidt, D.: Rainbow, a new multivariable polynomial signature scheme. In: Ioannidis, J., Keromytis, A., Yung, M. (eds.) ACNS 05. LNCS, vol. 3531, pp. 164–175. Springer, Heidelberg (Jun 2005)
12. Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., Stehlé, D.: CRYSTALS-Dilithium: A lattice-based digital signature scheme. IACR TCHES 2018(1), 238–268 (2018), `https://tches.iacr.org/index.php/TCHES/article/view/839`
13. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO'86. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (Aug 1987)
14. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM Journal on Computing 17(2), 281–308 (1988)
15. Grassi, L., Rechberger, C., Rotaru, D., Scholl, P., Smart, N.P.: MPC-friendly symmetric key primitives. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) ACM CCS 2016. pp. 430–443. ACM Press (Oct 2016)
16. Hulsing, A., Bernstein, D.J., Dobraunig, C., Eichlseder, M., Fluhrer, S., Gazdag, S.L., Kampanakis, P., Kolbl, S., Lange, T., Lauridsen, M.M., Mendel, F., Niederhagen, R., Rechberger, C., Rijneveld, J., Schwabe, P., Aumasson, J.P.: SPHINCS+. Tech. rep., National Institute of Standards and Technology (2019), available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions`
17. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Zero-knowledge proofs from secure multiparty computation. SIAM Journal on Computing 39(3), 1121–1152 (2009)
18. Iwaniec, H., Kowalski, E.: Analytic number theory, vol. 53. American Mathematical Soc. (2004)
19. Katz, J., Kolesnikov, V., Wang, X.: Improved non-interactive zero knowledge with applications to post-quantum signatures. In: Lie, D., Mannan, M., Backes, M., Wang, X. (eds.) ACM CCS 2018. pp. 525–537. ACM Press (Oct 2018)
20. Khovratovich, D.: Key recovery attacks on the legendre prfs within the birthday bound. Cryptology ePrint Archive, Report 2019/862 (2019), `https://eprint.iacr.org/2019/862`
21. Lyubashevsky, V., Ducas, L., Kiltz, E., Lepoint, T., Schwabe, P., Seiler, G., Stehlé, D.: CRYSTALS-DILITHIUM. Tech. rep., National Institute of Standards and Technology (2019), available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions`
22. National Institute of Standards and Technology: Post-quantum cryptography project (2016), https://csrc.nist.gov/projects/post-quantum-cryptography

23. Prest, T., Fouque, P.A., Hoffstein, J., Kirchner, P., Lyubashevsky, V., Pornin, T., Ricosset, T., Seiler, G., Whyte, W., Zhang, Z.: FALCON. Tech. rep., National Institute of Standards and Technology (2019), available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions`
24. Delpech de Saint Guilhem, C., De Meyer, L., Orsini, E., Smart, N.P.: BBQ: Using AES in picnic signatures. Cryptology ePrint Archive, Report 2019/781 (2019), `https://eprint.iacr.org/2019/781`
25. Samardjiska, S., Chen, M.S., Hulsing, A., Rijneveld, J., Schwabe, P.: MQDSS. Tech. rep., National Institute of Standards and Technology (2019), available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions`
26. Shor, P.W.: Algorithms for quantum computation: Discrete logarithms and factoring. In: Proceedings 35th annual symposium on foundations of computer science. pp. 124–134. Ieee (1994)
27. The Picnic team: The picnic signature algorithm specification (2019), https://github.com/microsoft/Picnic/blob/master/spec/spec-v2.1.pdf

A Proof of theorem 1

We will use the following version of the Weil bound for character sums [18].

Theorem 3. *Let p be a prime and χ a non-trivial multiplicative character of $\mathbb{F}_p^\times$ of order $d > 1$. If $f \in \mathbb{F}_p[X]$ has m distinct roots and is not a d-th power, then*

$$\left| \sum_{x \in \mathbb{F}_p} \chi\left(f(x)\right) \right| \leq (m-1)\sqrt{p}\,.$$

The following lemma immediately follows:

Lemma 1. *Let p be a prime and $k \,|\, p-1$. For any $K \neq K' \in \mathbb{F}_p$ and $a \in \mathbb{Z}_k$, let $I_{K,K',a}$ be the set of indices i such that $\mathcal{L}^k(K+i) = \mathcal{L}^k(K'+i) + a$. Then we have*

$$\frac{p}{k} - \sqrt{p} - 1 \leq \#I_{K,K',a} \leq \frac{p}{k} + \sqrt{p} + 2\,.$$

Proof. Let $\chi : \mathbb{F}_p^\times \to \mathbb{Z}_p$ be the restriction of $\mathcal{L}^k$ to $\mathbb{F}^\times$. Note that (unlike $\mathcal{L}^k$) χ is a group homomorphism. Define $f(i) = (i+K)(i+K')^{k-1}$ and let $\phi(a)$

be the number of i such that $i + K$ and $i + K'$ are non-zero and $\chi(f(i)) = a$. Clearly we have $\phi(a) \leq \#I_{K,K',a} \leq \phi(a) + 2$. Let $\hat{\phi} : \hat{\mathbb{Z}}_k \to \mathbb{C}$ be the fourier transform of ϕ. Then we have

$$\begin{aligned}\hat{\phi}(\rho) = \sum_{a \in \mathbb{Z}_k} \rho(a)\phi(a) = \sum_{a \in \mathbb{Z}_k} \rho(a) \sum_{i \in \mathbb{F}_p, i \neq K, i \neq K'} &\begin{cases} 1 \text{ if } \chi(f(i)) = a \\ 0 \text{ otherwise} \end{cases} \\ = \sum_{i \in \mathbb{F}_p, i \neq K, i \neq K'} \rho \circ \chi(f(i)) &\end{aligned}$$

Observe that $\rho \circ \chi$ is a multiplicative character of $\mathbb{F}_p^\times$, and that $\rho \circ \chi$ is trivial if and only if ρ is trivial. Clearly $\hat{\phi}(1) = p - 2$, and for non-trivial ρ, the Weil bound says that $|\hat{\phi}(\rho)| \leq \sqrt{p}$. Therefore, if follows from the inverse Fourier trasnform formula that

$$\phi(a) = \frac{1}{|\mathbb{Z}_k|} \sum_{\rho \in \hat{\mathbb{Z}}_k} \rho(a)\hat{\phi}(\rho) \leq \frac{p-2}{k} + \frac{k-1}{k}\sqrt{p} \leq \frac{p}{k} + \sqrt{p}\,.$$

and similarly that $\frac{p}{k} - \sqrt{p} - 1 \leq \phi(a)$. □

Now we can prove Theorem 1.

Proof. Accurding to lemma 1, For any $K' \neq K$ and $a \in \mathbb{Z}_k$, for a uniformly random set of inputs $\mathcal{I}$, the distance $d_H(F^k_{\mathcal{I}}(K') + (a, \ldots, a), s)$ is distributed as $\mathcal{B}(L, 1 - \alpha)$, for some $\alpha \in [1/k - \frac{1}{\sqrt{p}} - \frac{1}{p}, 1/k + \frac{1}{\sqrt{p}} + \frac{2}{p}]$. Therefore, the probability that for a tuple (K', a) we have $d_H(F^k_{\mathcal{I}}(K') + (a, \ldots, a), s) \leq \beta L$ is at most

$$\Pr[\mathcal{B}(L, 1/k + \frac{1}{\sqrt{p} + 2/p}) > (1 - \beta)L]\,.$$

Since there exists only $(p - 1)k$ possibile values for (K', a), the probability that there exists a non-trivial witness for the β-relaxed relation is at most $\Pr[\mathcal{B}(L, 1/k + \frac{1}{\sqrt{p}+2/p}) > (1 - \beta)L](p - 1)k$. □

B Security proof

To prove Theorem 2, we first reduce the EUF-KO security to the β-approximate PRF relation (Lemma 2); we then reduce the EUF-CMA security to the EUF-KO security (Lemma 3). For two real random variables A, B, we write $A \prec B$ if for all $x \in (-\infty, +\infty)$ we have $\Pr[A > x] \leq \Pr[B > x]$.

Lemma 2 (EUF-KO security). *Let $\mathcal{H}_{\mathsf{sd}}, \mathcal{H}_1, \mathcal{H}_2$ and $\mathcal{H}_3$ be modeled as random oracles and fix a constant $\beta \in [0,1]$. If there exists a PPT adversary $\mathcal{A}$ that makes $q_{\mathsf{sd}}, q_1, q_2$ and q_3 queries to the respective oracles, then there exists a PPT $\mathcal{B}$ which, given $\mathsf{pk} = F_L^k(K)$ for a random $K \in \mathbb{F}_p$ outputs a β-approximate witness for pk with probability at least $\mathbf{Adv}_{\mathcal{A}}^{EUF\text{-}KO}(1^\lambda) - e(q_{\mathsf{sd}}, q_1, q_2, q_3)$, with*

$$e(q_{\mathsf{sd}}, q_1, q_2, q_3) = \frac{MN(q_{\mathsf{sd}} + q_1 + q_2 + q_3)^2}{2^{2\lambda}} + \Pr[X + Y + Z = M],$$

where $X = \max(X_1, \dots, X_{q_1}), Y = \max(Y_1, \dots, Y_{q_2})$ and $Z = \max(Z_1, \dots, Z_{q_3})$, the X_i are i.i.d as $\mathcal{B}(M, (1-\beta)^B)$, the Y_i are i.i.d. as $\mathcal{B}(M - X, \frac{2}{p})$ and the Z_i are i.i.d. as $\mathcal{B}(M - X - Y, \frac{1}{N})$.

Proof. The algoritm $\mathcal{B}$ receives a statement $s = F_L^k(K)$ and forwards it to $\mathcal{A}$ as pk. Then, $\mathcal{B}$ simulates the random oracles $\mathcal{H}_{\mathsf{sd}}, \mathcal{H}_1, \mathcal{H}_2$ and $\mathcal{H}_3$ by maintaining initially empty lists of querries $\mathcal{Q}_{\mathsf{sd}}, \mathcal{Q}_1, \mathcal{Q}_2, \mathcal{Q}_3$. Moreover, $\mathcal{B}$ keeps initially empty tables $\mathcal{T}_s, \mathcal{T}_i$ and $\mathcal{T}_o$ for shares, inputs, and openings. If $\mathcal{A}$ queries one of the random oracles on an input that it has queried before, $\mathcal{B}$ responds as before; otherwise $\mathcal{B}$ does the following:

– $\mathcal{H}_{\mathsf{sd}}$: On new input $(\mathsf{salt}, \mathsf{sd})$, $\mathcal{B}$ samples $x \xleftarrow{\$} \{0,1\}^{2\lambda}$. If $x \in \mathsf{Bad_H}$, then $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ adds x to $\mathsf{Bad_H}$, $((\mathsf{salt}, \mathsf{sd}), x)$ to $\mathcal{Q}_{\mathsf{sd}}$ and returns x.
– $\mathcal{H}_1$: On new input $Q = (m, \mathsf{salt}, \sigma_1)$, with $\sigma_1 = ((\mathsf{C}_{e,i})_{i \in [N]}, (s_e^{(j)})_{j \in [B]}, \Delta K_e, \Delta c_e)_{e \in [M]})$, then $\mathcal{B}$ adds $\mathsf{C}_{e,i}$ to $\mathsf{Bad_H}$ for all $e \in [M]$ and $i \in [N]$. For any $(e,i) \in [M] \times [N]$ for which there exist $\mathsf{sd}_{e,i}$ such that $((\mathsf{salt}, \mathsf{sd}_{e,i}), \mathsf{C}_{e,i}) \in \mathcal{Q}_{\mathsf{sd}}$ define

$$k_{e,i}, a_{e,i}, b_{e,i}, c_{e,i}, r_{e,i}^{(1)}, \cdots, r_{e,i}^{(B)} \leftarrow \mathsf{Expand}(\mathsf{sd}_{e,i}) \text{ for all } j \in [N]$$

and add $\mathcal{T}_s[Q, e, i] = (k_{e,i}, a_{e,i}, b_{e,i}, c_{e,i}, r_{e,i}^{(1)}, \dots, r_{e,i}^{(B)})_{j \in [N]}$. If $\mathcal{T}_s[Q, e, i]$ is defined for all $i \in [N]$ for some $e \in [M]$, then we define

$$(k_e, a_e, b_e, c_e, r_e^{(1)}, \dots, r_e^{(B)}) \leftarrow \sum_{i \in [N]} (k_{e,i}, a_{e_i}, b_{e,i}, c_{e,i}, r_{e,i}^{(1)}, \dots, r_{e,i}^{(B)})$$
$$(k_e, c_e) \leftarrow (k_e + \Delta k_e, c_e + \Delta c_e)$$

and add $\mathcal{T}_i[Q, e] = (k_{e,i}, a_{e_i}, b_{e,i}, c_{e,i}, r_{e,i}^{(1)}, \dots, r_{e,i}^{(B)})$. Finally, $\mathcal{B}$ samples $x \xleftarrow{\$} \{0,1\}^{2\lambda}$. If $x \in \mathsf{Bad_H}$ then abort. Otherwise, $\mathcal{B}$ adds (Q, x) to $\mathcal{Q}_1$ and x to $\mathsf{Bad_H}$ and returns x.
– $\mathcal{H}_2$: On new input $Q = (h_1, \sigma_2)$, where $\sigma_2 = (o_e^{(j)})_{e \in [M], j \in [B]}$, $\mathcal{B}$ adds h_1 to $\mathsf{Bad_H}$ and samples $x \xleftarrow{\$} \{0,1\}^{2\lambda}$. If $x \in \mathsf{Bad_H}$ then abort. Otherwise, $\mathcal{B}$

adds (Q, x) to $\mathcal{Q}_2$ and x to $\mathsf{Bad_H}$. If there exists $(Q_1, h_1) \in \mathcal{Q}_1$, then $\mathcal{B}$ does the following: let $(\epsilon_e, \lambda_e^{(1)}, \ldots, \lambda_e^{(B)})_{e\in[M]} \leftarrow \mathsf{Expand}(x)$. For each $e \in [M]$ such that $\mathcal{T}_i(Q_1, e)$ is defined, compute

$$\alpha_e = a_e + \epsilon_e k_e, \qquad \beta_e = b_e + \sum_{j\in[B]} \lambda_e^{(j)r_e^{(j)}} \text{ and}$$

$$\gamma_e = -c_e + \alpha_e b_e + \beta_e a_e + \epsilon_i \sum_{k\in[B]} \lambda_i^{(k)}(o_e^{(j)} - I_e^{(j)} r_e^{(j)})$$

and add $\mathcal{T}_o[Q_2, e] = (\alpha_e, \beta_e, \gamma_e)$. Finally $\mathcal{B}$ returns x.

– $\mathcal{H}_3$: On new input $Q = (h_2, \sigma_3)$, $\mathcal{B}$ adds h_2 to $\mathsf{Bad_H}$ and samples $x \xleftarrow{\$} \{0,1\}^{2\lambda}$. If $x \in \mathsf{Bad_H}$ then $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ adds (Q, x) to $\mathcal{Q}_3$, x to $\mathsf{Bad_H}$ and returns x.

When $\mathcal{A}$ terminates, $\mathcal{B}$ goes through $\mathcal{T}_i$ and for each $(K_e, \ldots) \in \mathcal{T}_i$, $\mathcal{B}$ checks if K_e is a β-approximate witness. If it is, then $\mathcal{B}$ outputs K_e. If no entry in $\mathcal{T}_i$ contains a witness, $\mathcal{B}$ outputs $\perp$. Clearly, if $\mathcal{A}$ runs in time T, then $\mathcal{B}$ runs in time $T + O(q_{\mathsf{sd}} + q_1 + q_2 + q_3)$.

In the rest of the proof, we show that if $\mathcal{A}$ wins the EUF-KO game with probability ϵ, then $\mathcal{B}$ outputs a β-approximate witness with probability at least $\epsilon - e(q_{\mathsf{sd}}, q_1, q_2, q_3)$ as defined in the statement of Lemma 2.

Cheating in the first phase. Let $(Q_{\mathsf{best}_1}, h_{\mathsf{best}_1}) \in \mathcal{Q}_1$ be the "best" query-response pair that $\mathcal{A}$ received from $\mathcal{H}_1$, by which we mean the pair that maximizes $\#\mathsf{G}_1((Q, h))$ over all $(Q, h) \in \mathcal{Q}_1$, where $\mathsf{G}_1(Q, h = \{I_e^{(j)}\}_{e\in[M], j\in[B]})$ is defined as the set of "good executions" $e \in [M]$ such that $\mathcal{T}_i(Q, e)$ is defined and

$$\mathcal{L}^k((K_e + I_e^{(j)})r_e^{(j)}) = s_e^{(j)} + \mathsf{pk}_{I_e^{(j)}} \text{ for all } j \in [B]. \tag{2}$$

We show that, if $\mathcal{B}$ outputs $\perp$, then the number of good indices is bounded. More precicely, we prove that $\#\mathsf{G}_1(\sigma_{\mathsf{best}_1}, h_{\mathsf{best}_1})|_\perp \prec X$, where X is as defined in the statement of Lemma 2.

Indeed, for each distinct query to $\mathcal{H}_1$ of the form $Q = (m, \mathsf{salt}, \sigma_1)$, with $\sigma_1 = ((\mathsf{C}_{e,i})_{i\in[N]}, (s_e^{(j)})_{j\in[B]}, \Delta K_e, \Delta c_e)_{e\in[M]})$ and for all $e \in [M]$, let $\beta_e^{(j)}(Q) = d_H(F_L^k(K_e) + (\mathcal{L}^k(r_e^{(j)}), \ldots, \mathcal{L}^k(r_e^{(j)})), s_i^{(j)} + \mathsf{pk})$ if $\mathcal{T}_i(Q, e)$ is defined and $\beta_e^{(j)}(Q) = 1$ otherwise. The event $\perp$ implies that none of the K_e in $\mathcal{T}_i$ is a β-approximate witness, which means that $\beta_e^{(j)}(Q) > \beta$ for all $Q \in \mathcal{Q}_1, e \in [M]$ and $j \in [B]$.

Since the response $h = \{I_e^{(j)}\}_{e \in [M], j \in [B]}$ is uniform, the probability that for a certain e, Equation (2) holds is $\prod_{k \in [B]}(1 - \beta_i^{(k)}) \leq (1-\beta)^B$. Therefore, we have that $\#\mathsf{G}_1(Q,h)|_\perp \prec X_Q$, where $X_Q \sim \mathcal{B}(M,(1-\beta)^B)$. Finally, since $\mathsf{G}_1(Q_{\mathsf{best}_1}, h_{\mathsf{best}_1})$ is the maximum over at most q_1 values of $\mathsf{G}_1(Q,h)$, it follows that $\#\mathsf{G}_1(Q_{\mathsf{best}_1}, h_{\mathsf{best}_1})|_\perp \prec X$, with X as in the statement of Lemma 2.

Cheating in the second round. We now look at the best query-response pair $(Q_{\mathsf{best}_2}, h_{\mathsf{best}_2})$ that $\mathcal{A}$ received from $\mathcal{H}_2$. This is the pair for which $\#\mathsf{G}_2(Q_2,h_2)$ is maximum, where $\mathsf{G}_2(Q_2 = (h_1, (o_e^{(j)})_{e \in [M], j \in [B]}), h_2)$ is the set of "good" executions defined as follows: if there exists no Q_1, such that $(Q_1,h_1) \in \mathcal{Q}_1$, then all indices are bad (because this query can not lead to a valid signature). Otherwise, let $Q_1 = (m, \mathsf{salt}, ((\mathsf{C}_{e,i})_{i \in [N]}, (s_e^{(j)})_{j \in [B]}, \mathit{\Delta} K_e, \mathit{\Delta} c_e)_{e \in [M]}))$. If there exist $(e,j) \in [M] \times [B]$ such that

$$\mathcal{L}^k(o_e^{(j)}) \neq s_s^{(j)} + \mathsf{pk}_{I_e^{(j)}}, \tag{3}$$

then this query can also not result in a valid signature, so we define $\mathsf{G}_2(Q_2,h_2) = \{\}$. Otherwise, we say $\mathsf{G}_2(Q_2,h_2)$ is the set of executions $e \in [M]$ for which $\mathcal{T}_o[Q_2,e] = (\alpha_e, \beta_e, \gamma_e)$ is defined and such that $\alpha_e\beta_e = \gamma_e$.

Again, we prove that in the case that $\mathcal{B}$ outputs $\perp$, the number of good indices is bounded: $\#\mathsf{G}_2(Q_{best_2}, h_{best_2})|_\perp \prec X + Y$, where Y is defined as in the statement of Lemma 2.

Note that for fixed $a_e, b_e, c_e, K_e, r_e^{(1)}, \ldots, r_e^{(B)}$ and $o_e^{(1)}, \ldots, o_e^{(B)}$ the function $\alpha_e(\epsilon_e)\beta_e(\lambda_e^{(j)}) - \gamma_e(\epsilon_e, \lambda_e^{(j)})$ is a quadratic polynomial in $\epsilon_e, \lambda_e^{(1)}, \ldots, \lambda_e^{(B)}$. Moreover, this is the zero-polynomial if and only if $c_e = a_e b_e$ and $o_e^{(j)} = (K_e + I_e^{(j)}) r_e^{(j)}$ for all $j \in [B]$.

Let $Q = (h_1, \{o_e^{(j)}\}_{e \in [M], j \in [B]})$ be a query to $\mathcal{H}_2$. If there exists no $(Q_1,h_1) \in \mathcal{Q}_1$ then $\mathsf{G}_2(Q,h_2) = \{\}$ with probability 1. Otherwise, either $e \notin \mathsf{G}_1(\sigma_1,h_1)$, then either $o_e^{(j)} = (K_e + I_e^{(j)}) r_e^{(j)}$ for all $(e,j) \in [M] \times [B]$, in which case Equation (3) does not hold, so $\mathsf{G}_2(Q,h_2) = \{\}$ with probability 1, or $o_e^{(j)} \neq (K_e + I_e^{(j)}) r_e^{(j)}$ for some $j \in [B]$ in which case $\alpha_e\beta_e - \gamma_e$ is a non-zero quadratic polynomial in ϵ_e and $\lambda_e^{(j)}$, so the Schwartz-Zippel lemma says that for a uniformly random choice of $h_2 = \{\epsilon_e, \lambda_e^{(j)}\}_{e \in [M], j \in [B]} \in \mathbb{F}_p^{M(1+B)}$ the probability that $e \in \mathsf{G}_2(Q_2,h_2)$ is at most $2/p$. Therefore, we have that $\#\mathsf{G}_2(\sigma_2,h_2)|_{\#\mathsf{G}_1(\sigma_1,h_1)=M_1'} \prec M_1 + Y_Q'$, where $Y_q' \sim \mathcal{B}(M - M_1', 2/p)$. Since for integers $a \leq b$ and $p \in [0,1]$ we have $\mathcal{B}(b,p) \prec a + \mathcal{B}(b-a,p)$, this implies that $\#\mathsf{G}_2(\sigma_2,h_2)|_{\#\mathsf{G}_1(\mathsf{state}_{\mathsf{best},1})=M_1} \prec M_1 + Y_Q$, where $Y_Q \sim \mathcal{B}(M - M_1, 2/p)$. Since $\#\mathsf{G}_2(\mathsf{state}_{\mathsf{best},2})$ is the maximum over at most q_2

values of $\#\mathsf{G}_2(\mathsf{state})$ it follows that $\#\mathsf{G}_2(\mathsf{state}_{\mathsf{best},2})|_{M_1=\#\mathsf{G}_1(\mathsf{state}_{\mathsf{best},1})} \prec M_1+Y$. Finally, by conditioning on $\perp$ and summing over all M_1, we get

$$\#\mathsf{G}_2(\mathsf{state}_{best,2})|_\perp \prec \#\mathsf{G}_1(\mathsf{state}_{best,1})|_\perp + Y \prec X+Y.$$

Cheating in the third round. Finally, we can bound the probability that $\mathcal{A}$ wins the EUF-KO game, conditioned on $\mathcal{B}$ outputting $\perp$. Without loss of generality, we can assume that $\mathcal{A}$ outputs a signature σ such that, if Q_1, Q_2 and Q_3 are the queries that the verifier makes to $\mathcal{H}_1, \mathcal{H}_2$ and $\mathcal{H}_3$ to verify σ, then $\mathcal{A}$ has made these queries as well. (If this is not the case, then we can define $\mathcal{A}'$ that only outputs a signature after running the verification algorithm on $\mathcal{A}$'s output.) Now, for each query $Q = (h_2, (\{\alpha_e, \beta_e\}_{e\in M}, \{\alpha_{e,i}, \beta_{e,i}, \gamma_{e,i}\}_{e\in[M],i\in[N]}))$ that $\mathcal{A}$ makes to $\mathcal{H}_3$, we study the probability that this leads $\mathcal{A}$ to win the EUF-KO game. If there does not exist $Q' = (o_e^{(j)})_{e\in[M],j\in[B]}$ such that $(Q', h_2) \in \mathcal{Q}_2$ then this query cannot result in a win for $\mathcal{A}$, because $\mathcal{A}$ would need to find such a Q' at a later point, and $\mathcal{B}$ would abort if this happens. Take $e \in [M] \setminus \mathsf{G}_2(Q', h_2)$, then either $e \notin \mathsf{G}_2(Q', h_2)$ because there exists $(e', j) \in [M] \times [B]$ such that $\ell^k o_{e'}^{(j)} \neq s_{e'}^{(j)} + \mathsf{pk}_{I_{e'}^{(j)}}$, in which case, independent of h_3, σ_4, we have that $\mathsf{Vf}(\sigma) = 0$. Or otherwise $e \notin \mathsf{G}_2(Q', h_2)$ because α_e, β_e and γ_e are not defined or $\alpha_e\beta_e \neq \gamma_e$. In this case, the query can only result in a win if exactly $N-1$ of the parties "behave honestly" in the MPC protocol. By this we mean that for exactly $N-1$ values of $i \in [N]$ we have that there exists $\mathsf{sd}_{e,i}$ such that $(\mathsf{sd}_{e,i}, \mathsf{C}_{e,i}) \in \mathcal{Q}_{\mathsf{sd}}$ and, if we put $K_{e,i}, a_{e,i}, b_{e,i}, c_{e,i}, \{r_{e,i}^{(j)}\}_{j\in[B]} = \mathsf{Expand}(\mathsf{sd}_{e,i})$, then

$$\alpha_{e,i} = a_{e,i} + \epsilon_e K_{e,i}, \qquad \beta_{e,i} = b_{e,i} + \sum_k \lambda_e^{(j)} r_{e,i}^{(j)},$$
$$\gamma_{e,i} = -c_{e,i} + \alpha_e b_{e,i} + \beta_e a_{e,i} + \epsilon_e \sum_{j\in[B]} \lambda_e^{(j)} (o_e^{(j)} - I_e^{(j)} r_{e,i}^{(j)}).$$

Indeed, if there are less than $N-1$ honest parties, σ_4 cannot reveal $N-1$ honest views. In contrast if all the N parties act honestly, then we have $\gamma_e \neq \alpha_e\beta_e$, so the signature verification will also fail. The state $(\sigma_1, h_1, \sigma_2, h_2, \sigma_3)$ can only result in a win if $h_3 = \{\bar{i}_e\}_{e\in N}$ is such that $\bar{i}_e$ is the index of the dishonest party. Since $h_3 \in [N]^M$ is chosen uniformly at random, the probability that this happens for all the $e \notin \mathsf{G}_2(Q, h_3)$ is

$$\left(\frac{1}{N}\right)^{M-\#\mathsf{G}_2(Q',h_2)} \leq \left(\frac{1}{N}\right)^{M-\#\mathsf{G}_2(Q_{\mathsf{best},2},h_{\mathsf{best},2})}.$$

The probability that this happens for at least one of the at most q_3 queries is

$$\Pr[\mathcal{A}\,\mathsf{Wins}|\#\mathsf{G}_2(\mathsf{state}_{best,2}) = M_2] \leq 1 - \left(1 - \left(\frac{1}{N}\right)^{M-M_2}\right)^{q_3}.$$

Conditioning on $\mathcal{B}$ outputting $\perp$ and summing over all values of M_2 yields

$$\Pr[\mathcal{A}\,\mathsf{Wins}\,|\,\perp] \leq \Pr[X+Y+Z=M]\,.$$

To conclude. We now show that if $\mathcal{A}$ wins the EUF-KO game with probability ϵ, then $\mathcal{B}$ outputs a β-approximate witness with probability $\epsilon - e(q_{\mathsf{sd}}, q_1, q_2, q_3)$. Indeed, $\mathcal{B}$ either aborts outputs $\perp$ or outputs a β-approximate witness. The reduction $\mathcal{B}$ only aborts if one of the random oracles outputs one of the at most $q_{\mathsf{sd}} + MNq_1 + q_2 + q_3$ bad values. Therefore, we have

$$\Pr[\,\mathcal{E} \text{ aborts }] \leq \frac{MN(q_{\mathsf{sd}} + q_1 + q_2 + q_3)^2}{2^{2\lambda}}.$$

By the law of total probability we have

$$\begin{aligned}\Pr[\mathcal{A} \text{ wins}] =\ & \Pr[\mathcal{A} \text{ wins} \wedge \mathcal{B} \text{ aborts}] + \Pr[\mathcal{A} \text{ wins} \wedge \perp] \\ & + \Pr[\mathcal{A} \text{ wins} \wedge \mathcal{B} \text{ outputs witness}] \\ \leq\ & \Pr[\mathcal{B} \text{ aborts}] + \Pr[\mathcal{A} \text{ wins } |\perp] + \Pr[\mathcal{B} \text{ outputs witness}] \\ \leq\ & e(q_{\mathsf{sd}}, q_1, q_2, q_3) + \Pr[\mathcal{B} \text{ outputs witness}].\end{aligned}$$

Lemma 3. *Modeling the commitment scheme as a random oracle, if there is an adversary $\mathcal{A}$ that wins the EUF-CMA security game against LegRoast with advantage ϵ, then there exists an adversary $\mathcal{B}$ that, given oracle access to $\mathcal{A}$, and with a constant overhead factor, wins the EUF-KO security game agains LegRoast with probability at least $\epsilon - \frac{q_s(q_s+q_3)}{2^{2\lambda}} - \frac{q_{\mathsf{sd}}}{2^{\lambda}}$, where q_s, q_{sd} and q_3 are the number of queries that $\mathcal{A}$ makes to the signing oracle, $\mathcal{H}_{\mathsf{sd}}$ and $\mathcal{H}_3$ respectively.*

Proof. Let $\mathcal{A}$ be an adversary against the EUF-CMA security of LegRoast, we construct an adversary $\mathcal{B}$ against its EUF-KO security. When $\mathcal{B}$ is run on input pk, it starts $\mathcal{A}$ also on input pk. We first describe how $\mathcal{B}$ deals with random oracle queries and signature queries, then argue that its signature simulations are indistinguishable from real ones, and finally show that EUF-KO security implies EUF-CMA security.

Simulating random oracles. For each random oracle $\mathcal{B}$ maintains a table of input output pairs. When $\mathcal{A}$ queries one of the random oracles, $\mathcal{B}$ first checks if that query has been made before. If this is the case, $\mathcal{B}$ responds to $\mathcal{A}$ with the corresponding recorded output. If not, $\mathcal{B}$ returns a uniformly random output and records the new input-output pair in the table.

Signing oracle simulation. When $\mathcal{A}$ queries the signing oracle, $\mathcal{B}$ simulates a signature σ by sampling a random witness and cheating in the MPC verification phase to hide the fact it has sampled the witness as random. It then programs the last random oracle to always hide the party for which it has cheated. Formally, $\mathcal{B}$ simulates the signing oracle as follows:

1. To simulate σ_1, $\mathcal{B}$ follows Phase 1 as in the scheme with one difference: For each $e \in [M]$, it samples ΔK_e uniformly, effectively sampling K_e at random. $\mathcal{B}$ aborts if it picked a salt that was used in one of the earlier simulated signatures.
2. $\mathcal{B}$ simulates the random oracle to obtain $h_1 \leftarrow \mathcal{H}_1(m, \mathsf{salt}, \sigma_1)$.
3. To simulate σ_2, $\mathcal{B}$ samples $o_e^{(j)} \in \mathbb{F}_p^*$ for each $j \in [B]$ and $e \in [M]$ in such a way that $\mathcal{L}^k(o_e^{(j)}) - s_e^{(j)} = \mathsf{pk}_{I_e^{(j)}}$.
4. $\mathcal{B}$ simulates the random oracle to obtain $h_2 \leftarrow \mathcal{H}_2(h_1, \sigma_2)$.
5. To simulate σ_3, $\mathcal{B}$ must cheat during the sacrificing protocol to ensure that $\gamma_e = \alpha_e \beta_e$ for all executions. To do so, for each $e \in [M]$, $\mathcal{B}$ first samples $\bar{i}_e \in [N]$ at random. Then it computes Phase 5 honestly except for $\gamma_{e,\bar{i}_e}$; for that value, it instead sets $\gamma_{e,\bar{i}_e} \leftarrow \alpha_e \beta_e - \sum_{i \neq \bar{i}_e} \gamma_{e,i}$. Finally it sets σ_3 as in the scheme using the alternative $\gamma_{e,\bar{i}_e}$ value.
6. If (h_2, σ_3) has already been queried to $\mathcal{H}_3$, then $\mathcal{B}$ aborts. If not, $\mathcal{B}$ sets $h_3 = (\bar{i}_1, \ldots, \bar{i}_M)$ with the values it sampled previously and then programs its own random oracle $\mathcal{H}_3$ such that $h_3 \leftarrow \mathcal{H}_3(h_2, \sigma_3)$.
7. $\mathcal{B}$ follows the scheme to simulate σ_4 and the final signature σ.

Finally, when $\mathcal{A}$ outputs a forgery for its EUF-CMA game, $\mathcal{B}$ forwards it as its forgery for the EUF-KO game.

Simulation indistinguishability. If $\mathcal{B}$ doesn't abort, the simulation of the random oracles is perfect. Moreover, if $\mathcal{B}$ doesn't abort we show that $\mathcal{A}$'s can only distinguish a real signing oracle from the simulated oracle with advantage $q_{\mathsf{sd}}/2^\lambda$, where q_{sd} is the number of queries to $\mathcal{H}_{\mathsf{sd}}$.

The simulated signatures follow the exact same distribution as genuine signatures, with the only exception that in a genuine signature the $(\mathsf{C}_{e,\bar{i}_e})_{e \in [m]}$ are equal to $\mathcal{H}_{\mathsf{sd}}(\mathsf{salt}, e, \bar{i}_e, \mathsf{sd}_{e,\bar{i}_e)}$ for a value of $\mathsf{sd}_{e,\bar{i}_e}$ that expands to a consistent view of a party in the MPC protocol, whereas in the simulated case, $\mathsf{sd}_{e,\bar{i}_e}$ expands to the view of a cheating party. Since $\mathcal{H}_{\mathsf{sd}}$ is modelled as a random oracle, each of the $q_s \cdot M$ values of $\mathsf{C}_{e,\bar{i}_e}$ that $\mathcal{A}$ gets to see is just a random value, uncorrelated with the rest of the view of $\mathcal{A}$, *unless $\mathcal{A}$ has querried $\mathcal{H}_{\mathsf{sd}}$ on* $(\mathsf{salt}, e, \bar{i}_e, \mathsf{sd}_{e,\bar{i}_e})$. Since the $(\mathsf{salt}, e, \bar{i}_e)$ is unique per commitment ($\mathcal{B}$ aborts if a salt is repeated) and each seed has λ bits of min-entropy each query that $\mathcal{A}$ makes to $\mathcal{H}_{\mathsf{sd}}$ has a probability of at most $2^{-\lambda}$ of distinguishing the simulated signature oracle form a genuine signing oracle. Therefore, an adversary that makes q_{sd} queries to $\mathcal{H}_{\mathsf{sd}}$ has a distinguishing advantage bounded by $q_{\mathsf{sd}}/2^{\lambda}$.

EUF-KO security implies EUF-CMA security. Finally, we establish $\mathcal{B}$'s advantage against the EUF-KO security game. There are two moments at which $\mathcal{B}$ could abort: In phase 1 if a salt is repeated which happens with probability bounded by $q_s^2/2^{2\lambda}$ (recall that a salt consists of 2λ random bits) and in phase 6, if $\mathcal{B}$ fails to program the oracle $\mathcal{H}_3$, which happens with probability bounded by $q_s q_3/2^{2\lambda}$, since h_2 has 2λ bits of min entropy. Therefore, we have $\Pr[\mathcal{B} \text{ aborts}] \leq \frac{q_s(q_s+q_3)}{2^{2\lambda}}$, where q_s and q_3 denotes the number of signing queries and queries to $\mathcal{H}_3$ made by $\mathcal{A}$ respectively. Conditional on $\mathcal{B}$ not aborting, replacing the genuine oracles for the simulated oracles decreases the winning probability of $\mathcal{A}$ by at most $q_{\mathsf{sd}}/2^{\lambda}$. Therefore, given that the winning conditions for the EUF-KO and EUF-CMA games are identical, we have:

$$\mathbf{Adv}_{\mathcal{B}}^{\text{EUF-KO}}(1^\lambda) \geq \mathbf{Adv}_{\mathcal{A}}^{\text{EUF-CMA}}(1^\lambda) - \frac{q_s(q_s+q_3)}{2^{2\lambda}} - \frac{q_{\mathsf{sd}}}{2^{\lambda}}\,.$$

Chapter 9

Cryptanalysis of WalnutDSA

'Tis but a scratch!

– The Black Knight, Monty Python and the Holy Grail

Publication Data

Beullens W., Blackburn S.R. Practical Attacks Against the Walnut Digital Signature Scheme. In: Advances in Cryptology – ASIACRYPT 2018. Lecture Notes in Computer Science, vol 11272, pages 35-61. Springer, 2018.

My contribution

Main author. I found the attacks that made it into the paper, and did most of the writing. I wrote the implementation of the attacks and did the benchmarking.

Practical attacks against the Walnut digital signature scheme

Ward Beullens[1] and Simon R. Blackburn[2]

[1] imec-COSIC KU Leuven,
Kasteelpark Arenberg 10 - bus 2452, 3001 Heverlee, Belgium
Ward.Beullens@esat.kuleuven.be

[2] Department of Mathematics, Royal Holloway, University of London,
Egham, Surrey TW20 0EX, Royal Holloway, UK
S.Blackburn@rhul.ac.uk

Abstract. Recently, NIST started the process of standardizing quantum-resistant public-key cryptographic algorithms. WalnutDSA, the subject of this paper, is one of the 20 proposed signature schemes that are being considered for standardization. Walnut relies on a one-way function called E-Multiplication, which has a rich algebraic structure. This paper shows that this structure can be exploited to launch several practical attacks against the Walnut cryptosystem. The attacks work very well in practice; it is possible to forge signatures and compute equivalent secret keys for the 128-bit and 256-bit security parameters submitted to NIST in less than a second and in less than a minute respectively.

Keywords: WalnutDSA, NIST PQC, post-quantum digital signatures, cryptanalysis, group based cryptography

1 Introduction

As more and more progress is being made towards building large scale quantum computers, the need for cryptography that can withstand cryptanalysis from these machines has become increasingly urgent. In recognition of this fact, NIST has started the Post-Quantum Cryptography standardization project [20] and made a call for quantum-resistant public-key cryptographic algorithms for standardization. The community has answered this call by submitting 20 proposals for signature schemes and 49 proposals for encryption schemes. One of the submitted signature schemes is the Walnut digital signature algorithm [5, 8], submitted by D. Atkins and owned by SecureRF. SecureRF is a corporation founded in 2004 that develops and licenses public-key security tools for the low-resource processors powering the Internet of Things (IoT) [1]. SecureRF received the ARM Techcon 2017 "Best contribution to IoT security" award for the Walnut signature scheme and their "Key Agreement Protocol". SecureRF wants to achieve widespread usage of the Walnut signature scheme in the booming IoT market through standardization, partnerships with manufactures like Intel and STMicroelectronics and by providing free toolkits for popular low end platforms. Because of this potential for widespread use, it is crucial to analyze the Walnut scheme for potential weaknesses.

Related work. For its security, Walnut relies on problems taken from the theory of infinite non-commutative groups (more precisely, problems based on an action of a braid group on a finite set via the coloured Burau representation). The idea of using infinite groups in cryptography goes back at least as far as Wagner and Magyarik [26] in 1985; see González Vasco and Steinwandt [25] for an attack on this proposal. Problems in braid groups have been proposed as hard problems for cryptographic primitives for about 20 years now: key agreement protocols due to Ko et al. [18] and Anshel, Anshel and Goldfeld [3] (which is in a more general setting) are the best known examples. The Algebraic Eraser [4] is a more recent proposal, also promoted by SecureRF, which uses many of the same algebraic techniques as Walnut. Early cryptanalyses of these schemes used length-based attacks [15, 16], but the most convincing attacks [11, 10, 12, 17, 23] have generally been based on representation theory (where 'linearisation' techniques reduce the underlying security to a problem in linear algebra). Walnut is interesting because these linearisation techniques do not seem to apply.

The first attack on (an earlier version of) Walnut [6] is due to Hart et al. [14]. The attack forges signatures in minutes for the suggested parameters, but the resulting signatures are significantly longer than legitimately produced signatures. So the Hart et al. attack can be blocked by imposing a length limit on valid signatures. In their submission to NIST, the designers of Walnut impose such a length limit in order to block the Hart et al. attack, but also modify the scheme in a significant way (in particular changing the form of the public and private keys) in an attempt to block the attack altogether.

Contributions. In this paper we present three independent practical attacks on the Walnut signature scheme. The first attack is a modification of the attack of [14] that applies to the adapted version of Walnut that was supposed to resist this attack. This first attack is practical, but has the same limitation as the original attack by Hart et al: the forged signatures are very long. This attack demonstrates that the modifications intended to completely block the Hart et al. attack are not effective, but the attack can be blocked (as before) by imposing a length limit on signatures. The other two attacks presented in this paper produce forgeries whose lengths are the same or even shorter than those of legitimate signatures. The second attack in this paper constructs pairs of messages with the same signature; the attacker can choose a large amount of the structure of these messages. Our third attack directly constructs equivalent secret keys. We are able to forge signatures and compute equivalent secret keys in under one second for 128-bit security parameters, and in less than a minute for 256-bit security parameters. This shows that the parameter sets submitted to the NIST PQC standardization project are totally insecure, and that the corresponding implementation (which was freely available on the SecureRF website before we notified them of our attacks) should not be used. Our attacks exploit various algebraic properties of the one-way function called E-Multiplication, which is fundamental for the Walnut scheme (and other SecureRF methods). In fact, we give a practical algorithm to break the one-wayness of this function for the parameters submitted to NIST. In order to avoid the attacks given here, the parameters of Walnut need to be increased significantly (see the conclusion at the end of the paper for details). However, with these increased parameter sizes, it seems that Walnut loses its performance advantage over other post-quantum signature schemes such as lattice-based, code-based, multivariate and hash-based signatures.

Outline. In Sect. 2 we explain some necessary preliminaries such as distinguished point collision finding, a very short introduction to braid groups, and an explanation of E-Multiplication and the workings of the Walnut signature scheme. The following sections, Sections 3, 4 and 5, each introduce a practical attack against the Walnut scheme and discusses the feasibility of countermeasures. Sect. 3 contains an adaptation of the factorization attack of [14] that applies to the updated version of Walnut that was submitted to NIST. Sect. 4 describes an attack where we use a generic distinguished point collision finding method to find two documents d_1 and d_2 such that a signature that is valid for d_1 is automatically valid for d_2 and vice versa. In Sect. 5 we give an algorithm that breaks the one-wayness of the E-Multiplication map. This algorithm can be used to forge signatures and compute equivalent secret keys, even for the 256 bits of security parameters. The last section presents the conclusions of the paper.

2 Preliminaries

2.1 Distinguished point collision finding

The attacks introduced in this paper rely on a collision finding algorithm that is able to find a collision in any function $f : D \to D$ which maps a domain D to itself. Our algorithm of choice is the distinguished point method of van Oorschot and Wiener [24]. Finding a single collision with this method has the same $O(\sqrt{|D|})$ time complexity as Pollard's rho method with cycle finding [21, 22], but it can be parallelized more efficiently. Moreover, the method of van Oorschot and Wiener is much more efficient for finding multiple collisions; the number of collisions found grows quadratically with the time spent.

The algorithm repeatedly chooses a random starting point $x_1 \in D$ and iteratively applies the function f to obtain a chain of values $x_1, x_2, \cdots$, where $x_i = f(x_{i-1})$ for all $i > 1$. This process continues until a *distinguished* value x_k is reached. This is a value which satisfies some easily verified property, such as having a fixed number of leading zero bits. This property is chosen such that it is satisfied by a fraction ϑ of the elements of D. When the distinguished point is reached the starting point x_1, the distinguished point x_k and the length k of the chain is stored in a table. Assuming f behaves like a random function, after an expected number of $O(\sqrt{|D|})$ function calls the current chain will collide with one of the previously computed chains. From this point on we will follow the same chain and we will end up at the same distinguished point. We read the starting poins x_1, x'_1 and the corresponding chain lengths k, k' from the table. Without loss of generality, we assume that $k \geq k'$. We then know that for some $i < k'$

$$x_{k-k'+i} \neq x'_i \qquad \text{and} \qquad f(x_{k-k'+i}) = f(x'_i)\,,$$

unless the starting point x'_0 appears in the chain starting at x_0 (which only happens with a very small probability). This collision can be extracted with $k - k' + 2i$ function calls. If we require more than one collision we can continue the process, maintaining the contents of the table. Since over time the table will contain more and more chains, the rate at which collisions are found will also increase.

2.2 Braid groups

Informally, the braid group on N strands is a group whose elements are represented by a configuration of N non-intersecting vertical strands in three dimensional space, where 2 configurations are considered equal if one can be transformed continuously into the other configuration without intersecting the strands. The group multiplication is defined as the concatenation of the strands. E. Artin [9] showed that there is an equivalent definition of braid groups, given by the presentation

$$\left\langle b_1, \cdots, b_{N-1} \,\middle|\, \begin{array}{cc} b_i b_j = b_j b_i & \text{for } 1 \le i < j < N \text{ and } j - i \ge 2 \\ b_i b_{i+1} b_i = b_{i+1} b_i b_{i+1} & \text{for } 1 \le i < N-1 \end{array} \right\rangle .$$

Here, the *Artin generator* b_i represents the braid where the i-th strand crosses over the $(i+1)$-th strand. The relations $b_i b_j = b_j b_i$ for $|i-j| \ge 2$ correspond to the fact that crossings that involve different strands are free to move past each other. The relations $b_i b_{i+1} b_i = b_{i+1} b_i b_{i+1}$ correspond to moving one strand over the crossing of two other strands. The Artin generators and their relations are graphically represented in Fig. 1, 2 and 3.

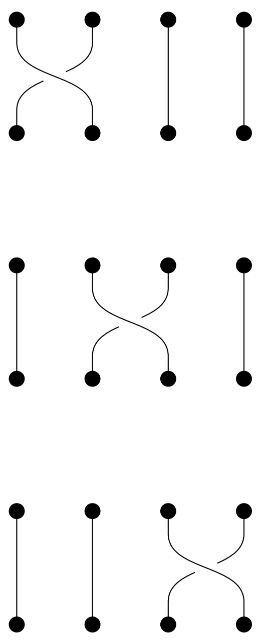


Fig. 1. The three Artin generators b_1, b_2 and b_3 that generate B_4.

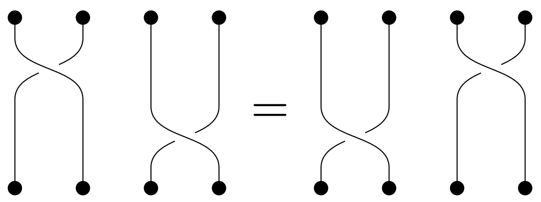


Fig. 2. Crossings that do not share strands commute, i.e. $b_1 b_3 = b_3 b_1$

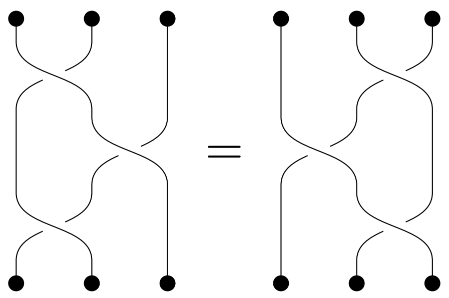


Fig. 3. The first strand moves over the crossing of strand 2 and 3, i.e. $b_1 b_2 b_1 = b_2 b_1 b_2$.

There is a natural homomorphism $\sigma : B_N \to S_N$ from the braid group on N strands to the symmetric group of order N that maps each braid to the permutation obtained by following the strands. This map sends an Artin generator b_i to the transposition $\sigma(b_i) = (i \quad i+1)$. Elements in the kernel of this homomorphism are called *pure braids*, the kernel itself is called the *pure braid group on N strands* and is denoted by P_N.

The braid group B_2 on two strands is the infinite cyclic group, so this group is its own center. For $N > 2$ the center of the braid group on N strands is generated by the full-twist braid which is obtained by grabbing the ends of the strands of the identity braid and rotating them by 360 degrees [13]. This braid is commonly denoted by Δ^2 and is depicted in Fig. 4.

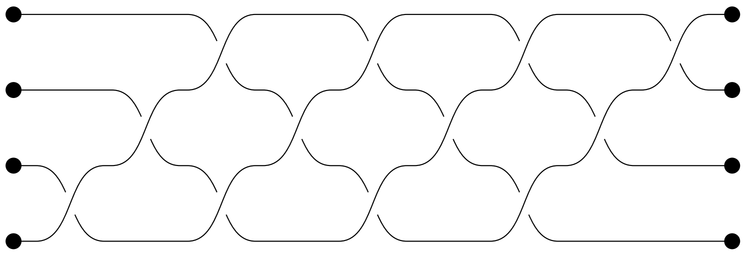


Fig. 4. The full-twist braid Δ^2 in the braid group on 4 strands.

2.3 The colored Burau representation and E-multiplication

The Walnut digital signature algorithm relies heavily on a group action called *E-Multiplication.* To define this group action we need the colored Burau Representation (see, for example, Anshel et al. [2]) which is a homomorphism from the braid group B_N to the *colored Burau group* $GL_N(\mathbb{Z}[t_1^{\pm 1}, \cdots, t_N^{\pm 1}]) \rtimes S_N$. This group is defined as a semidirect product, by letting the symmetric group S_N act on $GL_N(\mathbb{Z}[t_1^{\pm 1}, \cdots, t_N^{\pm 1}])$ by permuting the variables t_i. More concretely, the elements of the colored Burau group are pairs $(\mathbf{A}(t_1, \cdots, t_N), \pi)$ where $\pi \in S_N$ is a permutation and where $\mathbf{A}(t_1, \cdots, t_N)$ is an invertible $N \times N$ matrix whose entries lie in $\mathbb{Z}[t_1^{\pm 1}, \cdots, t_N^{\pm 1}]$. Multiplication in the colored Burau group is defined by

$$(\mathbf{A}(t_1, \cdots, t_N), \pi) \cdot (\mathbf{B}(t_1, \cdots, t_N), \tau) := (\mathbf{A}(t_1, \cdots, t_N) \cdot \pi(\mathbf{B}(t_1, \cdots, t_N)), \pi\tau)$$

$$= (\mathbf{A}(t_1, \cdots, t_N) \cdot \mathbf{B}(t_{\pi(1)}, \cdots, t_{\pi(N)}), \pi\tau).$$

The *colored Burau representation* $CB : B_N \to GL_N(\mathbb{Z}[t_1^{\pm1}, \cdots, t_N^{\pm1}]) \rtimes S_N$ is defined at each Artin generator as $CB(b_i) = (CBM(i), \sigma(b_i))$, where $CBM(i)$ is a matrix and $\sigma(b_i)$ is a permutation, defined as follows. The permutation $\sigma(b_i)$ is the transposition $(i\ i+1)$. We define $CBM(b_1)$, the colored Burau matrix of b_1, as

$$CBM(b_1) = \left(\begin{array}{cc|c} -t_1 & 1 & 0 \\ 0 & 1 & 0 \\ \hline 0 & 0 & \mathbb{1}_{N-2} \end{array}\right),$$

where $\mathbb{1}_{N-2}$ is the $(N-2) \times (N-2)$ identity matrix. For $i > 1$ the colored Burau matrix of b_i is defined as

$$CBM(b_i) = \left(\begin{array}{c|ccc|c} \mathbb{1}_{i-2} & 0 & 0 & 0 & 0 \\ \hline 0 & 1 & 0 & 0 & 0 \\ 0 & t_i & -t_i & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 0 & \mathbb{1}_{N-i-1} \end{array}\right).$$

This definition of $CB(b_i)$ is compatible with the relations of the braid group, so it can be extended to define a homomorphism on the entire group B_N. For a braid b, the matrix component of $CB(b)$ is called the *colored Burau matrix* of b and is denoted by $CBM(b)$, the permutation component of $CB(b)$ is simply equal to $\sigma(b)$. This implies that pure braids are mapped into the subgroup $GL_N(\mathbb{Z}[t_1^{\pm1}, \cdots, t_N^{\pm1}]) \subset GL_N(\mathbb{Z}[t_1^{\pm1}, \cdots, t_N^{\pm1}]) \rtimes S_N$.

Now we fix a finite field $\mathbb{F}_q$, and for any integer k with $1 < k \leq N$ we define A_k to be the group of invertible N-by-N matrices of the form

$$A_k = \left\{ \begin{pmatrix} X & Y & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \mathbb{1}_{N-k} \end{pmatrix} | X \in GL_{k-1}(\mathbb{F}_q), Y \in \mathbb{F}_q^{k-1} \right\}.$$

Let $T = [\tau_1, \cdots, \tau_N]$ be a list of N non-zero values in a finite field $\mathbb{F}_q$. The *evaluation* $M \downarrow_T$ of a matrix $\mathbf{M}(t_1, \cdots, t_n)$ at T is computed by replacing each occurence of variable t_i by τ:

$$M \downarrow_T := M(\tau_1, \cdots, \tau_N).$$

This map is a well-defined homomorphism on the image $\mathrm{im}(CB)$ of CB. For a list T containing N non-zero finite field elements, we can now define a right group action, called E-Multiplication and denoted by $\star$, of the braid group B_N on the set $A_N \times S_N$. A braid b acts on the first component of the pair (M, π) by multiplying from the right with a matrix obtained from the colored Burau matrix of b by permuting the variables t_i using π and then evaluating at T. The second component of the action is obtained by multiplying on the right by $\sigma(b)$. Written out symbolically, this is

$$(M, \pi) \star b := (M \cdot \pi(CBM(b)) \downarrow_T, \pi\sigma(b)) .$$

The fact that this defines a group action follows from the fact that the colored Burau representation is a homomorphism of groups. In practice, when calculating $(M, \pi) \star b$, the action is calculated one Artin generator at a time (see Alg. 1). Given the sparsity of the colored Burau matrices $CBM(b_i)$, acting with an Artin generator requires only a few column operations on M and one swap on π, so this is very efficient. This action was first introduced in [4], where it was used to build a key agreement protocol. More recently, E-Multiplication has been used as the basic building block for a cryptographic hash function [7] and the Walnut digital signature scheme [5].

```
Algorithm E-Multiplication
input: $(M, \pi)$ — a pair in $A_N \times S_N$ to act on
       $s$ — a braid to act with
       $T = \{\tau_1, \cdots, \tau_N\}$ — a list of T-values
output: $(M, \pi)$ — the resulting pair

1: while $|s| > 0$ do
2:     $b_i^{\pm 1} || s \leftarrow s$        ▷ split the first generator $b_i^{\pm 1}$ from the rest of $s$
3:     $N \leftarrow CBM(b_i)^{\pm 1}$        ▷ The CB Matrix of $b_i$, inverted if necessary.
4:     $N \leftarrow N(\tau_{\pi(i)})$        ▷ Evaluate in $\tau_{\pi(i)}$
5:     $M \leftarrow M \cdot N$
6:     $\pi \leftarrow \pi \circ \sigma(b_i)$
7: end while
8: return $(M, \pi)$
```

Alg. 1. The algorithm for computing the E-Multiplication action.

By letting B_N act on $(\mathbb{1}_N, e) \in A_N \times S_N$ we define a map $\mathcal{P}$

$$\mathcal{P} : B_N \to A_N \times S_N : s \mapsto (\mathbb{1}_N, e) \star s .$$

When restricted to the subgroup of pure braids P_N, the second component of $\mathcal{P}$ always maps to the identity permutation, so we can think of it as a map $\mathcal{P}|_{P_N} : P_N \to A_N$. The map $\mathcal{P}|_{P_N}$ is actually a homomorphism because it is the composition of the colored Burau representation $CB : P_N \to GL_N(\mathbb{Z}[t_1^{\pm 1}, \cdots, t_N^{\pm 1}])$ and the evaluation homomorphism $|_T : \mathrm{im}(CB) \to A_N$. Moreover, if we further restrict $\mathcal{P}$ to the subgroup P_k of pure braids where only the first k strands cross over each other, i.e. the intersection of P_N with the subgroup generated by $b_1, \cdots, b_{k-1}$, the homomorphism $\mathcal{P}|_{P_k} : P_k \to A_k$ maps into the subgroup A_k. This fact will be exploited in the attack of Sect. 5.

2.4 The Walnut signature scheme

We now introduce the Walnut signature scheme, which is the subject of our cryptanalysis. Before we describe the key generation, signing and verification algorithms (Alg. 2, 4 and 5) in detail we will summarize the scheme very briefly: the secret key consists of two braids s_1, s_2 and the public key is $(M_1, \pi_1) = \mathcal{P}(s_1)$ and $M_2 = \mathsf{mat}(\mathcal{P}(s_2))$, the matrix component of $\mathcal{P}(s_2)$. To sign or verify a document d it is hashed and encoded as a pure braid $E(d)$ with an encoding mechanism E. The Walnut design [5] defines a braid sig to be a valid signature for the document d if and only if the verification equation

$$\mathsf{mat}(\mathcal{P}(s_1) \star \mathsf{sig}) = \mathsf{mat}(\mathcal{P}(E(d))) \cdot M_2 \tag{1}$$

is satisfied. However, this equation is equal to the matrix component of

$$\mathcal{P}(s_1) \star \mathsf{sig} = \mathcal{P}(E(d)) \star s_2 \,, \tag{2}$$

and the permutation component of equation (2) is also satisfied by all the legitimately produced signatures. In this document we define a valid signature as a braid sig that satisfies the stronger verification equation (2). It is clear that $\mathsf{sig} = s_1^{-1} E(d) s_2$ would be a valid signature. In order to prevent length-based attacks [16, 19] *cloaking elements*, namely braids that do not affect E-Multiplication, are inserted into the signature and the braids are put through a rewriting algorithm so that (it is hoped) s_1 and s_2 cannot easily be extracted from the signature.

Parameters. The scheme is parametrized by:

- The number N of strands of the braid group that is being used (which is equal to the dimension of the associated square matrices).
- The size q of a finite field $\mathbb{F}_q$.
- A rewriting algorithm $\mathcal{R} : B_N \to B_N$.
- L and l, the length of certain random braid words.
- A hash function $\mathcal{H}$.

[3] Signatures have variable length. The reported signature size is an average, using the BKL + Dehornoy rewriting method.

Table 1. The Walnut parameter sets submitted to the NIST Post Quantum Cryptography project, and the corresponding public key and signature sizes.

claimed security level	128-bit	256-bit
N	8	8
q	2^5	2^8
L	15	30
l	132	287
$\mathcal{H}$	SHA2-256	SHA2-512
Public key length	83 Bytes	128 Bytes
Signature length[3]	≈646 Bytes	≈ 1248 Bytes

Key generation. The private key consists of two randomly chosen braids $s_1, s_2 \in B_N$ of length l. The braids are chosen such that their underlying permutations $\sigma(s_1)$ and $\sigma(s_2)$ are distinct and not equal to the identity permutation e. The public key contains a list $T = \{\tau_1 = 1, \tau_2 = 1, \tau_3, \cdots, \tau_n\} \in \mathbb{F}_q^N$ of N elements of the finite field $\mathbb{F}_q$ such that the first two elements are equal to 1, and such that the remaining values are non-zero and different from 1. The public key also contains $\mathcal{P}(s_1)$ and the matrix component of $\mathcal{P}(s_2)$.

```
Algorithm GenerateKeys
input: random bits to generate $s_1, s_2$ and $\tau_i$
output: pk — a public key
        sk — a corresponding secret key
1: $s_1, s_2 \leftarrow$ a randomly chosen braid words of length $l$.
2: $\tau_1, \tau_2 \leftarrow 1$
3: for $i$ from 3 to $N$ do
4:     $\tau_i \leftarrow$ a randomly chosen field element, not equal to 0 or 1
5: end for
6: $T \leftarrow \{\tau_1, \cdots, \tau_N\}$
7: $(M_1, \pi_1) \leftarrow \mathcal{P}(s_1)$
8: $(M_2, \pi_2) \leftarrow \mathcal{P}(s_2)$
9: return pk $= (T, M_1, M_2, \pi_1)$ and sk $= (s_1, s_2)$
```

Alg. 2. The Walnut key pair generation algorithm

Encoding a document. In order to sign a document d or verify a signature the document is converted to a pure braid $E(d) \in P_N$. This conversion consists of two stages. First, a hash digest of d is computed with a standard hash function (SHA2-256 or SHA2-512), then this hash is converted to a braid. To make the second conversion 4 pure braids g_1, g_2, g_3, g_4 are fixed such that they generate a free subgroup of P_N. The Walnut specification document [8] defines

$$\begin{aligned} g_1 &= b_N b_{N-1} \cdots b_2 \cdot b_1^2 \cdot b_2^{-1} \cdots b_{N-1}^{-1} b_N^{-1} \\ g_2 &= b_N b_{N-1} \cdots b_4 \cdot b_3^2 \cdot b_4^{-1} \cdots b_{N-1}^{-1} b_N^{-1} \\ g_3 &= b_N b_{N-1} \cdots b_6 \cdot b_5^2 \cdot b_6^{-1} \cdots b_{N-1}^{-1} b_N^{-1} \\ g_4 &= b_N b_{N-1} \cdots b_8 \cdot b_7^2 \cdot b_8^{-1} \cdots b_{N-1}^{-1} b_N^{-1} \,. \end{aligned}$$

The encoding process starts from the trivial braid. Two bits are taken from the hash digest to choose one g_i of the 4 generators, and the next two bits of the digest define an exponent $e \in \{1, 2, 3, 4\}$. Then g_i^e is appended to the braid, and four bits are removed from the digest. This is repeated until the entire hash output is consumed.

```
Algorithm EncodeDocument
input: A document $d$
output: $b$ — a pure braid
1: $\mathsf{h} \leftarrow \mathcal{H}(d)$
2: $b \leftarrow e$
3: for $a$ from 0 to $|\mathsf{h}|/4 - 1$ do
4:     $\mathsf{i} \leftarrow \mathsf{h}[4a : 4a + 1]$                ▷ Select index
5:     $\mathsf{e} \leftarrow \mathsf{h}[4a + 2 : 4a + 3]+1$          ▷ Select exponent
6:     $b \leftarrow b \cdot g_{\mathsf{i}}^{\mathsf{e}}$
7: end for
8: return $b$
```

Alg. 3. The document encoding mechanism.

Signing algorithm. The signing algorithm produces a signature which is a braid word of the form

$$\mathsf{sig}' = v_1 \cdot s_1^{-1} \cdot v \cdot E(d) \cdot s_2 \cdot v_2 \,,$$

where v_1, v and v_2 are so called cloaking elements, which are braids in the stabilizer of $\mathcal{P}(s_1), (\mathbb{1}_N, e)$ and $\mathcal{P}(E(d)s_2)$ respectively. Therefore we have

$$\begin{aligned}(\mathbb{1}_N, e) \star s_1 \cdot \mathsf{sig}' &= \mathcal{P}(s_1) \star s_1^{-1} \cdot v \cdot E(d) \cdot s_2 \cdot v_2 \\ &= (\mathbb{1}_N, e) \star v \cdot E(d) \cdot s_2 \cdot v_2 \\ &= \mathcal{P}(E(d)s_2) \cdot v_2 \\ &= (\mathbb{1}_N, e) \star E(d) \cdot s_2 \,,\end{aligned}$$

so sig' is a valid signature. To hide the secret key s_1 and s_2 which are substrings of sig' one of three proposed rewriting algorithms (BKL + Dehornoy, Stochastic + Dehornoy or Stochastic) is used to produce a different braid word sig which represents the same braid as sig'. The various rewriting algorithms differ in performance and in the length of the signatures that are produced.

The cloaking elements are generated using the following lemma.

Lemma 1. *Suppose that $\tau_1 = \tau_2 = 1$. Take any pair $(M, \pi) \in A_N \times S_N$, an Artin generator b_i, and any braid w such that*

$$\pi \circ \sigma(w)(i) = 1 \qquad \textit{and} \qquad \pi \circ \sigma(w)(i+1) = 2\,.$$

Then the braid $v = w \cdot b_i^2 \cdot w^{-1}$ is in the stabilizer of (M, π).

To produce a cloaking element for $\mathcal{P}(s_1), (\mathbb{1}_N, e)$ or $\mathcal{P}(E(d)s_2)$ we first pick a random integer i such that $1 < i < N$, then we choose a random braid w satisfying the conditions of Lemma 1 and we set $v = wb_i^2w^{-1}$. For the details of how w is chosen (which depends on the parameter L) and the details on how the various rewriting algorithms work we refer to the WalnutDSA NIST submission [8].

Verification Algorithm. Given a document d, a public key $\mathsf{pk} = (T, M_1, M_2, \pi_1)$ and a signature sig. The verification algorithm simply calculates the encoding of the message $E(d)$ and the matrix components of $(M_1, \pi_1) \star \mathsf{sig}$ and $\mathcal{P}(E(d))$. It then accepts the signature if the computed matrices satisfy the equation

$$\mathsf{mat}((M_1, \pi_1) \star \mathsf{sig}) = \mathsf{mat}(\mathcal{P}(E(d))) \cdot M_2\,.$$

```
Algorithm Sign
input: $d$ — a document to sign
       $\mathsf{sk} = (s_1, s_2)$ — a secret key
output: $\mathsf{sig}$ — a signature for document $d$
1: $v_1 \leftarrow$ GetCloakingElement($\sigma(s_1)$)
2: $v \leftarrow$ GetCloakingElement($e$)
3: $v_2 \leftarrow$ GetCloakingElement($\sigma(s_2)$)
4: $E_d \leftarrow$ EncodeDocument($d$)
5: $\mathsf{sig}' \leftarrow v_1 \cdot s_1^{-1} \cdot v \cdot E_d \cdot s_2 \cdot v_2$
6: $\mathsf{sig} \leftarrow \mathcal{R}(\mathsf{sig}')$
7: return $\mathsf{sig}$
```

Alg. 4. The Walnut signature generation algorithm

```
Algorithm Verify
input: $d$ — a document
       $\mathsf{pk} = (T, M_1, M_2, \pi_1)$ — a secret key
       $\mathsf{sig}$ — a signature
output: True if $\mathsf{sig}$ is a valid signature for $d$, False otherwise
1: $E_d \leftarrow$EncodeDocument($d$)
2: LHS $\leftarrow \mathsf{mat}($ E-Multiplication($(M_1, \pi_1), \mathsf{sig}, T$))
3: RHS $\leftarrow \mathsf{mat}($ E-Multiplication($(\mathbb{1}_N, e), E_d, T$)) $\cdot M_2$
4: if LHS equals RHS then
5:     return True
6: end if
7: return False
```

Alg. 5. The Walnut signature verification algorithm

3 A factorization attack

This section describes an adaptation of the factorization attack of Hart et al. [14] on an earlier version of Walnut [6]. This earlier version is a special case of the newer construction where the two secret braids s_1 and s_2 are equal. This means that the secret key essentially consists of only a single braid s, and that the public key is a single matrix-permutation pair $(M, \pi) = \mathcal{P}(s)$. The signing and verification algorithms of the earlier version are the same as the algorithms described in the previous section after substituting s for s_1 and s_2, and substituting M for M_1 and M_2. The attack of Hart et al. exploits the following malleability property:

Theorem 1. (for the earlier version of Walnut with $s_1 = s_2$) *Suppose d, d_1, d_2 are three documents. Let h, h_1, h_2 be the matrix part of $\mathcal{P}(E(d)), \mathcal{P}(E(d_1))$ and $\mathcal{P}(E(d_2))$ respectively. Then we have*

1. *If $h = h_1^{-1}$ and sig_1 is a valid signature for d_1, then sig_1^{-1} is a valid signature for d.*
2. *If $h = h_1 \cdot h_2$ and $\mathsf{sig}_1, \mathsf{sig}_2$ are valid signatures for d_1 and d_2 respectively, then $\mathsf{sig}_1\mathsf{sig}_2$ is a valid signature for d.*

This opens up the following strategy to attack the signature scheme. First we collect a set of valid document-signature pairs (d_i, sig_i) and we let $h_i = \mathsf{mat}(\mathcal{P}(E(d_i)))$. Then, if we want to forge a signature for a document d with $h = \mathsf{mat}(\mathcal{P}(E(d)))$ it suffices to write h as a product $\prod_{j=1}^{k} h_{i_j}^{e_j}$ of the h_i. Once we have this, a valid signature for d is given by $\prod_{j=1}^{k} \mathsf{sig}_{i_j}^{e_j}$. This reduces breaking the signature scheme to breaking the factorization problem in A_N:

Factorization problem in a group G. Given a list of elements $g_1, \cdots, g_k$ that generate the group G and a target element g, write the target g as a (preferably short) product of the g_i and their inverses.

The paper of Hart et al [14] proposes an algorithm to solve the factorization problem in A_N, exploiting a chain of subgroups. This allows them to forge signatures in minutes, but the factorizations that are found by the algorithm are very long, so this results in very long signatures. The forged signatures are many orders of magnitude longer than legitimate signatures, so the Walnut scheme can be saved by imposing an upper limit to the length of the signatures.

The Walnut signature scheme was adapted to destroy the malleability property of Theorem 1. In the remainder of this section we prove that an adapted version of the maleability property still holds for the new WalnutDSA scheme and we show how the property can be used to reduce breaking Walnut to solving the factorization problem in A_N, which can be solved with the techniques of [14].

3.1 Signature malleability of Walnut

Walnut has the following malleability property, which is a generalization of the property discovered by Hart et al. (Theorem 1).

Theorem 2. *Suppose d, d_1, d_2 are three documents. Let h, h_1, h_2 be the matrix part of $\mathcal{P}(E(d)), \mathcal{P}(E(d_1))$ and $\mathcal{P}(E(d_2))$ respectively. Let $s_1, s_2, s_3 \in B_N$ be three braids. Then*

1. *If $h = h_1^{-1}$ and sig_1 is a valid signature for d_1 under the public key $(\mathcal{P}(s_1), \mathcal{P}(s_2))$, then sig_1^{-1} is a valid signature for d under the public key $(\mathcal{P}(s_2), \mathcal{P}(s_1))$.*
2. *If $h = h_1 \cdot h_2$ and $\mathsf{sig}_1, \mathsf{sig}_2$ are valid signatures for d_1 and d_2 under the public keys $(\mathcal{P}(s_1), \mathcal{P}(s_2))$ and $(\mathcal{P}(s_2), \mathcal{P}(s_3))$ respectively, then $\mathsf{sig}_1 \cdot \mathsf{sig}_2$ is a valid signature for d under the public key $(\mathcal{P}(s_1), \mathcal{P}(s_3))$.*

Proof. We start by proving 1. Since sig_1 is a valid signature for d_1 we have

$$\mathcal{P}(s_1) \star \mathsf{sig}_1 = \mathcal{P}(E(d_1)) \star s_2.$$

Acting on this by sig_1^{-1} and using the definition of $\mathcal{P}$ we get

$$(\mathbb{1}_N, e) \star s_1 = (h_1, e) \star s_2 \cdot \mathsf{sig}_1^{-1},$$

where we have used the fact that $E(d_1)$ is a pure braid. Multiplying the matrix part of this equality by h_1^{-1} from the left (multiplying on the left by a matrix commutes with $\star$), we get

$$(h_1^{-1}, e) \star s_1 = (\mathbb{1}_N, e) \star s_2 \cdot \mathsf{sig}_1^{-1},$$

or equivalently

$$\mathcal{P}(E(d)) \star s_1 = \mathcal{P}(s_2) \star \mathsf{sig}_1^{-1},$$

which shows that sig_1^{-1} is a valid signature for d for the public key $(\mathcal{P}(s_2), \mathcal{P}(s_1))$.

To prove 2 we start by acting with sig_2 on the verification equation for sig_1 to get

$$\begin{aligned} \mathcal{P}(s_1) \star \mathsf{sig}_1 \cdot \mathsf{sig}_2 &= \mathcal{P}(E(d_1)) \star s_2 \cdot \mathsf{sig}_2 \\ &= (h_1 \cdot CBM(s_2)_{\downarrow T} \cdot^{\sigma(s_2)} (CBM(\mathsf{sig}_2))_{\downarrow T}, \sigma(s_2) \circ \sigma(\mathsf{sig}_2)) . \end{aligned}$$

Using the fact that sig_2 is a valid signature for d_2 under the public key $(\mathcal{P}(s_2), \mathcal{P}(s_3)$ we see that

$$\begin{aligned} \mathcal{P}(s_1) \star \mathsf{sig}_1 \cdot \mathsf{sig}_2 &= (h_1 \cdot h_2 \cdot CBM(s_3)_{\downarrow T}, \sigma(s_3)) \\ &= (h_1 \cdot h_2, e) \star s_3 \\ &= \mathcal{P}(E(d)) \star s_3 , \end{aligned}$$

which shows that $\mathsf{sig}_1 \cdot \mathsf{sig}_2$ is a valid signature for d under the public key $(\mathcal{P}(s_1), \mathcal{P}(s_3))$.

3.2 The factorization attack

Given an oracle $\mathcal{O}_f$ (which can be instantiated by the algorithm of [14]) that solves the factorization for the group A_N, we can now break Walnut as follows. Suppose we want to forge a signature for a document d under the public key $(\mathcal{P}(s_1), \mathcal{P}(s_2))$. Let h be the matrix part of $\mathcal{P}(E(d))$. We start by collecting a number of document-signature pairs $(d_1, \mathsf{sig}_1), \cdots, (d_k, \mathsf{sig}_k)$ that are valid under the same public key, and we compute the matrix part h_i of each pair $\mathcal{P}(E(d_i))$. Now it suffices to find a factorization $h = h_{i_1} \cdot h_{i_2}^{-1} \cdot h_{i_3} \cdots h_{i_{m-1}}^{-1} \cdot h_{i_m}$ whose factors have powers that alternate between 1 and -1. Indeed, combining properties of Theorem 2 we see that $\mathsf{sig}_{i_1} \cdot \mathsf{sig}_{i_2}^{-1}$ is a valid signature for any document d' such that $\mathsf{mat}(\mathcal{P}(E(d'))) = h_{i_1} \cdot h_{i_2}^{-1}$ under the public key $(\mathcal{P}(s_1), \mathcal{P}(s_1))$. Adding an extra factor, we get that $\mathsf{sig}_{i_1} \cdot \mathsf{sig}_{i_2}^{-1} \cdot \mathsf{sig}_{i_3}$ is a valid signature for an appropriate document under the public key $(\mathcal{P}(s_1), \mathcal{P}(s_2))$. Continuing the same argument for the odd number m of factors of the product we get that $\mathsf{sig}_{i_1} \cdot \mathsf{sig}_{i_2}^{-1} \cdot \mathsf{sig}_{i_3} \cdots \mathsf{sig}_{i_{m-1}}^{-1} \cdot \mathsf{sig}_{i_m}$ is a valid document for d under the desired public key $(\mathcal{P}(s_1), \mathcal{P}(s_2))$.

We can use the oracle $\mathcal{O}_f$ to find the factorization $h = h_{i_1} \cdot h_{i_2}^{-1} \cdot h_{i_3} \cdots h_{i_{m-1}}^{-1} \cdot h_{i_m}$. We construct the list of generators

$$\mathsf{gens} = \{h_i \cdot h_j^{-1} \quad | \quad i \neq j \in \{1, \cdots, k\}\}$$

and call the oracle $\mathcal{O}_f$ to obtain a factorization for $h \cdot h_1^{-1}$ with factors in this set of generators. Appending the factor h_1 to the resulting factorization we then get a factorization of h of the desired form.

3.3 Implications and countermeasures

The factorization algorithm of [14] has a time complexity of $O\left(q^{\frac{N-1}{2}}\right)$ and for the 128 bit security parameters of Walnut (i.e. $N = 8$, $q = 2^5$) the algorithm finds a factorization in minutes. However, these factorizations contain roughly 2^{25} factors, so the forged signatures are the concatenation of roughly 2^{25} legitimate signatures. This implies that the forged signatures are many orders of magnitude longer than legitimate signatures and so they can be detected easily by the verifier. To protect against this attack it suffices to impose a limit on the length of signatures. Interestingly, when the WalnutDSA scheme was updated to counter the attack of [14], no such upper limit was included in the design. Our adaptation of the attack shows that this limit is necessary for the security of the scheme, because long forgeries can be produced in a matter of minutes.

The implementation submitted to the NIST PQC standardization project implicitly imposes such an upper limit by specifying that the length of the signature (measured by the number of Artin generators) be encoded by two bytes. This effectively limits the signature braids to be at most 2^{16} Artin generators long. Therefore the attack cannot be used to break the NIST implementation of WalnutDSA.

4 A collision search attack

From the verification equation

$$\mathcal{P}(s_1) \star \mathsf{sig} = \mathcal{P}(E(d)) \star s_2$$

it is clear that the only dependence on the document d is through the encoding mechanism E and the mapping $\mathcal{P}$. This implies that if d_1 and d_2 are two documents such that $\mathcal{P}(E(d_1) = \mathcal{P}(E(d_2))$, then any signature that is valid for d_1 is automatically valid for d_2 and vice versa. Therefore breaking EUF-CMA security reduces to finding such a pair of documents. Once an attacker has found two such documents he can ask the signing oracle to produce a signature sig for d_1, and return (sig, d_2) to win the EUF-CMA game. Since the first step of the encoding function E is the application of a cryptographically secure hash function to the document d we cannot reasonably expect to have a more efficient way of finding collisions than with a generic collision search. A generic collision search requires roughly $|\mathcal{P}(E(\{0,1\}^*))|^{1/2}$ evaluations of $\mathcal{P} \circ E$. In the rest of this section we give an upper bound for this quantity and we demonstrate with computer experiments that a collision attack is practical.

4.1 Sizes of orbits of E-multiplication

To estimate the time complexity of the collision search attack we need to find the size of $\mathcal{P}(E(\{0,1\}^*))$. Without much motivation the designers of WalnutDSA claim that $q^{N(N-3)}N!$ is a conservative lower bound on the number values that $\mathcal{P}$ can take [5]. For 128-bit and 256-bit security parameters this number is roughly 2^{216} and 2^{336} respectively, which means that finding a collision should require roughly 2^{108} and 2^{168} evaluations of $\mathcal{P} \circ E$. Note that this is already significantly less than the claimed security levels. Moreover, an elementary analysis will reveal that this "conservative lower bound" is actually much larger than the true value of $|\mathcal{P}(B_N)|$. Even worse, when $\mathcal{P}$ is restricted to the set of braids that can be produced by the encoding mechanism E, the number of values that can be reached is much smaller still.

We know that $\mathcal{P}$, when restricted to the subgroup of pure braids, is a homomorphism from P_N to A_N. This implies that the full twist braid Δ^2 (see Sect. 2.2) which generates the center of P_N is mapped to a matrix in the center of $\mathcal{P}(P_N)$. It can be verified that the only matrix in the center of A_N is the identity matrix, but for a randomly chosen set of T-values $\mathcal{P}(\Delta^2)$ is typically not the identity matrix. This means that $\mathcal{P}(A_N)$ sits inside the centralizer of $\mathcal{P}(\Delta^2)$, which is typically a proper subspace of $\langle A_N \rangle$. This begs the question of what the dimension of $\langle \mathcal{P}(P_N) \rangle$ is. From computer experiments we can conclude that for randomly chosen T-values this is equal to the dimension of the centralizer of $\mathcal{P}(\Delta^2)$, which is equal to $(N-1)^2+1$ (since $\mathcal{P}(\Delta^2)$ has one eigenspace of dimension $N-1$ and one of dimension 1). However, if we impose the extra condition that the first two T-values are equal to one, $\mathcal{P}(P_N)$ is contained in an affine subspace of dimension $(N-2)^2+1$, so $|\mathcal{P}(P_N)|$ is at most $q^{(N-2)^2+1}$. Our computer experiments suggest that this upper bound is reasonably tight, and so we estimate $|\mathcal{P}(P_N)| \approx q^{(N-2)^2+1}$. Since P_N is a subgroup of B_N of index $N!$ we have $|\mathcal{P}(B_N)| < q^{(N-2)^2+1}N!$. Note that this upper bound is strictly lower than the lower bound which was claimed by the designers of Walnut.

Any braid output by the encoding mechanism E is a product of the generators g_1, g_2, g_3, g_4. From computer experiments we conclude that when applying $\mathcal{P}$ to braids of this form we end up with matrices in an affine subspace of surprisingly low dimension. We found that they live in a subspace of dimension 13, independent of the values of q or N (provided that $N^2 > 13$). This means that $|\mathcal{P}(E(\{0,1\}^*))|$ is at most q^{13}, and that finding a collision cannot take much more than $q^{13/2}$ evaluations of $\mathcal{P} \circ E$. For 128-bit security parameters this number is as low as $2^{32.5}$, and for 256-bit security parameters this is 2^{52}.

4.2 Implementation

We implemented the generic collision finding algorithm of van Oorschot and Wiener [24] (briefly explained in Sect. 2.1) and used it to find collisions for the function $g \circ \mathcal{P} \circ E$, where g is a function that takes the ouput of $\mathcal{P}$, and converts it to some plausible document d. Even though the method is completely generic, it is still efficient enough to find colliding documents in practice. It took approximately $2^{32.2}$ evaluations of f (which agrees very well with the expected value of $2^{32.5}$) or one hour on a standard desktop PC to find the following pair of colliding documents.

d_1 ="I would like to receive 9156659270109667494 free samples
of chocolate chip cookies."

d_2 ="I would like to receive 10213941738370235726 free samples
of gluten-free raisin cookies."

The documents can be cunningly crafted such that a victim would be eager to sign the first document with his/her secret key. However, by producing a signature for this document, the victim would unknowingly also sign the second document, which might lead to unsavory consequences.

4.3 Implications and countermeasures

This practical attack shows that the Walnut signature scheme should not be used with the parameters that are submitted to the NIST PQC project.

Increasing q to raise $q^{13/2}$ to the required security level would lead to $q = 2^{20}$ and $q = 2^{40}$ for 128-bit and 256-bit security parameters respectively. For 256-bit security parameters this would increase the size of the public key by a factor of 5 and we estimate that this would slow down the verification algorithm by a factor of 25. A better approach would be to change the encoding algorithm to output pure braids that are not restricted to the subgroup generated by g_1, g_2, g_3, g_4 (or any other proper subgroup). Since $\mathcal{P}(P_N)$ is contained in an affine subspace of dimension $(N-2)^2+1$, this would lead to an upper bound on the complexity of the attack of $\sqrt{q^{(N-2)^2+1}}$ evaluations of $\mathcal{P} \circ E$. We would then only need a slight increase in the parameters. For example, 256 bits of security would be achieved (against this attack) by the parameters $q = 2^8$ and $N = 10$, leading to an increase of the key size of roughly 50% and the signature size by at least 25%.

5 Reversing E-multiplication

A fundamental hard problem underlying the Walnut signature scheme is the "Reversing E-Multiplication" (REM) problem. This problem asks, given a pair $(M, \sigma) \in A_N \times S_N$, such that $(M, \sigma) = (\mathbb{1}_N, e) \star s$ for some braid $s \in B_N$, to find a braid $s' \in B_N$ such that $(\mathbb{1}_N, e) \star s' = (M, \sigma)$. In other words, the problem is to break the one-wayness of the function

$$\mathcal{P} : B_N \to A_N \times S_N : s \mapsto (\mathbb{1}_N, e) \star s.$$

The secret key in Walnut consists of two braids $s_1, s_2 \in B_N$. The corresponding public key is $\mathcal{P}(s_1)$ and the matrix part of $\mathcal{P}(s_2)$. The fact that the permutation part of $\mathcal{P}(s_2)$ is not available to the attacker is not a problem, because given a single signature sig which is valid for any message (which might be unknown to the attacker), the attacker can deduce the permutation of s_2 from the permutation component of the verification equation (2)

$$\sigma(s_1) \circ \sigma(\mathsf{sig}) = \sigma(s_2).$$

After solving the REM problem to get s_1', s_2' such that $\mathcal{P}(s_1) = \mathcal{P}(s_1')$ and $\mathcal{P}(s_2) = \mathcal{P}(s_2')$, an attacker can use the pair (s_1', s_2') as a secret key to sign any message. Alternatively, instead of solving two instances of the REM problem to obtain an equivalent secret key, it is also possible to solve a single instance of the REM problem to obtain a signature for a document which can be chosen freely.

In this section we give an algorithm that solves the REM problem in practice for the parameters that are proposed for Walnut. First, we describe a generic birthday attack that can reverse any group action. Then, we introduce an algorithm that exploits the subgroup structure of B_N and is much more efficient.

5.1 Birthday attack

A brute force attack would repeatedly pick a random $s \in B_N$, compute $(\mathbb{1}_N, e) \star s$ and check if this is equal to the target (M, σ). This attack would take $O(|\mathcal{P}(B_N)|)$ attempts, where $|\mathcal{P}(B_N)|$ is the size of the orbit of $(\mathbb{1}_N, e)$. A more efficient approach is to look for $s_1, s_2 \in B_N$ such that

$$(M, \sigma) \star s_1 = (\mathbb{1}_N, e) \star s_2 \,.$$

If such s_1 and s_2 are found, the solution to the REM problem is given by $s_2 s_1^{-1}$. A naive way of finding s_1 and s_2 is to compute a large table containing $\sqrt{|\mathcal{P}(B_N)|}$ values of s_1 and the corresponding values of $(M, \sigma) \star s_1$ and check for random values of s_2 whether $(\mathbb{1}_N, e) \star s_2$ lies in this table. This method takes $O(\sqrt{|\mathcal{P}(B_N)|})$ E-Multiplications, but requires a lot of memory. The problem can be reduced to collision finding for a function $f : \mathcal{P}(B_N) \to \mathcal{P}(B_N)$. Then, distinguished point methods (see Sect. 2.1) can solve the REM problem with the same time complexity as the naive approach but with constant memory complexity. Concretely, suppose $\mathsf{b} : \mathcal{P}(B_N) \to \{0, 1\}$ and $\mathsf{s} : \mathcal{P}(B_N) \to B_N$ are hash functions that take a matrix and a permutation from the orbit of $(\mathbb{1}_N, e)$ as input, and output a bit or a braid respectively. Then we can define

$$f(x) = \begin{cases} (\mathbb{1}_N, e) \star \mathsf{s}(x) \text{ if } \mathsf{b}(x) = 0, \\ (M, \sigma) \star \mathsf{s}(x) \text{ if } \mathsf{b}(x) = 1. \end{cases}$$

If s outputs sufficiently long braids such that $\mathcal{P}(\mathsf{s}(x))$ is distributed uniformly in the orbit of $(\mathbb{1}_N, e)$, then the distinguished point method will yield collisions $f(x_1) = f(x_2)$ such that $\mathsf{b}(x_1) \neq \mathsf{b}(x_2)$ with probability 1/2. Once such a collision is found, a solution to the REM problem is given by $\mathsf{s}(x_1)\mathsf{s}(x_2)^{-1}$ or $\mathsf{s}(x_2)\mathsf{s}(x_1)^{-1}$ when $\mathsf{b}(x_1)$ is 0 or 1 respectively. For the security parameters aiming for 128 bits of security, the size of the orbit $\mathcal{P}(B_N)$ is bounded by 2^{200} (see Sect. 4.1), so the number of E-multiplications required to solve REM is not much more than 2^{100}, considerably less than 2^{128} but still far from practical. For the 256 bit security parameters the number of E-multiplications is not much more than 2^{157}.

5.2 Subgroup chain attack

We next propose a practical method for solving the REM problem that improves the attack above by exploiting the following chain of subgroups of B_N:

$$\{e\} = P_1 \subset P_2 \subset \cdots \subset P_N \subset B_N.$$

The map $\mathcal{P}$ sends a braid to an element of $A_N \times S_N$ and, when restricted to P_i it is a homomorphism to A_i (see Sect. 2.3). Therefore we have the following commuting diagram:

$$\begin{tikzcd}
\{e\} \arrow[r, hook] \arrow[d, "\mathcal{P}"] & P_2 \arrow[r, hook] \arrow[d, "\mathcal{P}"] & \cdots \arrow[r, hook] & P_N \arrow[r, hook] \arrow[d, "\mathcal{P}"] & B_N \arrow[d, dashed, "\mathcal{P}"] \\
\{(\mathbb{1}_N, e)\} \arrow[r, hook] & A_2 \arrow[r, hook] & \cdots \arrow[r, hook] & A_N \arrow[r, hook] & A_N \times S_N
\end{tikzcd}$$

The meet-in-the-middle attacks in the previous subsection attempt to find a braid s such that $(M, \sigma)\star s = (\mathbb{1}_N, e)$ in one step. Given this subgroup structure, it is more efficient to solve REM in several steps. The first step is to find a braid $s' \in B_N$ such that $(M, \sigma) \star s' = (M', e) \in A_N$. This is trivial because any $s' \in B_N$ whose underlying permutation is σ^{-1} will do the job. The next step is to find a pure braid $s_N \in P_N$ such that $(M', e) \star s_N \in A_{N-1}$. Then, one continues iteratively to find $s_i \in P_i$ such that $(M, \sigma) \star s' s_N \cdots s_i \in A_{i-1}$. After the last step we have found $s' s_N \cdots s_2$ such that $(M, \sigma) \star s' s_N \cdots s_2 = (\mathbb{1}_N, e)$, so $(s' s_n \cdots s_2)^{-1}$ is a solution to the REM problem.

One caveat when using this method is that, a priori, it is possible to get stuck. After each step, we get a new target $(M, \sigma) \star s' s_N \cdots s_i$ which is sampled randomly from $\mathcal{P}(P_i) \cap A_{i-1}$. However, from that point on, we will only act on this target with pure braids from P_{i-1}. This means that if the new target is not in $\mathcal{P}(P_{i-1})$ we will not be able to complete the attack. If we assume for each i that

$$\mathcal{P}(P_i) \cap A_{i-1} = \mathcal{P}(P_{i-1}),$$

then the attack is guaranteed to work. In practice, this assumption seems to hold with large probability for the parameter sets that are proposed, because the algorithm works without having to backtrack. We encounter this problem when instantiating the Walnut scheme with a smaller finite field such as $\mathbb{F}_5$. Then, it occurs for a small but noticeable fraction of the choices of T-values that for some small i all the generators of $\mathcal{P}(P_{i-1})$ have determinant 1 or -1, while the subgroup $\mathcal{P}(P_i) \cap A_{i-1}$ contains matrices with any determinant. This problem is unlikely to occur in large finite fields and with large i, because then there are many generators of $\mathcal{P}(P_{i-1})$ that all have to map to a matrix with determinant ± 1.

Each step can be solved with a collision search in the space $A_{i-1}\mathcal{P}(P_i)\backslash A_{i-1}$ of cosets of A_{i-1} in $A_{i-1}\mathcal{P}(P_i)$. Let $\mathsf{b} : A_{i-1}\mathcal{P}(P_i)\backslash A_{i-1} \to \{0,1\}$ and $\mathsf{s} : A_{i-1}\mathcal{P}(P_i)\backslash A_{i-1} \to P_i$ be hash functions that take a right coset and output a bit or a pure braid respectively. Then we can define $f : A_{i-1}\mathcal{P}(P_i)\backslash A_{i-1} \to A_{i-1}\mathcal{P}(P_i)\backslash A_{i-1}$ as

$$f(x) = \begin{cases} A_{i-1}\mathcal{P}(\mathsf{s}(x)) & \text{if } \mathsf{b}(x) = 0, \\ A_{i-1}M'\mathcal{P}(s_N \cdots s_{i+1}\mathsf{s}(x)) & \text{if } \mathsf{b}(x) = 1. \end{cases}$$

The distinguished point method can find collisions $f(x_1) = f(x_2)$ at a cost of roughly $\sqrt{|A_{i-1}\mathcal{P}(P_i)\backslash A_{i-1}|}$ E-Multiplications. Under the assumption we made earlier that $\mathcal{P}(P_i) \cap A_{i-1} = \mathcal{P}(P_{i-1})$ this is equal to $\sqrt{|\mathcal{P}(P_i)|/|\mathcal{P}(P_{i-1})|}$ E-Multiplications.

If we plug the estimate of $|\mathcal{P}(P_i)| \approx q^{(i-2)^2+1}$ from Sect. 4.1 into this formula, we get an estimate of $\sqrt{\frac{q^{(i-2)^2+1}}{q^{(i-3)^2+1}}} = q^{i-5/2}$ E-Multiplications to find s_i. The runtime of the algorithm is dominated by the step that searches for s_N, which is estimated to require $q^{N-5/2}$ E-Multiplications. For 128-bit security parameters this number is $2^{27.5}$ and this agrees very well with our computer experiments. For 256-bit security parameters, the required number of E-Multiplications is estimated to be 2^{44}.

5.3 Representing and manipulating cosets of A_k.

In order to implement the hash functions b and s we need to be able to uniquely represent right cosets with respect to A_k. We give a method to do this efficiently in this subsection. Suppose, $\mathbf{X}, \mathbf{Y}$ are two matrices in A_N, that are in the same right coset of A_{N-1}. That is, there exists a matrix $\mathbf{A} \in A_{N-1}$ such that $\mathbf{AX} = \mathbf{Y}$. If we split up the matrices to make their structure visible we get:

$$\begin{pmatrix} \mathbf{A}_1 & \mathbf{A}_2 & \mathbf{0} \\ \mathbf{0} & 1 & 0 \\ \mathbf{0} & 0 & 1 \end{pmatrix} \begin{pmatrix} \mathbf{X}_1 & \mathbf{X}_2 \\ \mathbf{X}_3 & \mathbf{X}_4 \\ \mathbf{0} & 1 \end{pmatrix} = \begin{pmatrix} \mathbf{Y}_1 & \mathbf{Y}_2 \\ \mathbf{Y}_3 & \mathbf{Y}_4 \\ \mathbf{0} & 1 \end{pmatrix} .$$

From this it is obvious that the $(N-1)$-th row of $\mathbf{X}$ and $\mathbf{Y}$ are identical, and that the first $(N-1)$ rows of $\mathbf{X}$ and $\mathbf{Y}$ span the same $(N-1)$-dimensional subspace. It is easily checked that the converse also holds, which implies that the right coset of A_{N-1} that contains a matrix $\mathbf{X} \in A_N$ is completely determined by the $(N-1)$-th row of $\mathbf{X}$ and the subspace spanned by the first $N-1$ rows of $\mathbf{X}$. In turn, this subspace is uniquely represented by the row reduced echelon form of the upper $(N-1)$-by-N submatrix of $\mathbf{X}$, which will be of the form

$$\begin{pmatrix} \mathbb{I}_{N-1} & \mathbf{v} \end{pmatrix}$$

for some $\mathbf{v} \in \mathbb{F}^{N-1}$. Therefore, the coset containing $\mathbf{X}$ is completely determined by the $(N-1)$-th row of $\mathbf{X}$, and the last column of the first $N-1$ rows of X after putting it in row reduced echelon form. More generally, we have the following lemma.

Lemma 2. *A right coset of $A_k \backslash A_{k-1}$ with representative $\mathbf{X} \in A_k$ is completely determined by the pair of vectors $(\mathbf{v}_1, \mathbf{v}_2) \in \mathbb{F}_q^N \times \mathbb{F}_q^{k-1}$, where $\mathbf{v}_1$ is the $(k-1)$-th row of $\mathbf{X}$ and $\mathbf{v}_2$ is the k-th column of the matrix $\mathbf{X}'$, which is obtained from $\mathbf{X}$ by taking the first $k-1$ rows and putting them in row reduced echelon form.*

This lemma gives a method for deciding whether two matrices $\mathbf{X}$ and $\mathbf{Y}$ are in the same coset. One simply computes the pair of vectors for both matrices $\mathbf{X}$ and $\mathbf{Y}$ and checks whether they are equal. To run the algorithm we also need a way to act on cosets by multiplying on the right by matrices. One way to do this is to work with a representative from the coset and carry out a matrix multiplication to get a representative from the next coset. It is more efficient to compute directly with the two-vector representation of the coset. The following lemma gives a way to do this.

Lemma 3. *Suppose $\mathbf{M}$ is a matrix in A_k for some k with $1 < k \leq N$. Let $\mathbf{A} \in GL_{k-1}(\mathbb{F}_q)$ and $\mathbf{b} \in \mathbb{F}_q^{k-1}$ be submatrices of $\mathbf{M}$ such that*

$$\mathbf{M} = \begin{pmatrix} \mathbf{A} & \mathbf{b} & \mathbf{0} \\ \mathbf{0} & 1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbb{1}_{N-k} \end{pmatrix} .$$

If $(\mathbf{v}_1, \mathbf{v_2})$ *is the representation of a coset* S *as in Lemma 2, then the representation of the coset* $S\mathbf{M}$ *is given by* $(\mathbf{v}_1\mathbf{M}, \mathbf{A}^{-1}(\mathbf{b} + \mathbf{v}_2))$.

Proof. It is clear that if $\mathbf{v}_1$ is the $(k-1)$-th row of a representative of S, then $\mathbf{v}_1\mathbf{M}$ is the $(k-1)$-th row of a representative $S\mathbf{M}$. For the second vector, suppose that the subspace spanned by the first $k-1$ rows of a representative of S is the row subspace of

$$\begin{pmatrix} \mathbb{1} & \mathbf{v}_2 & \mathbf{0} \end{pmatrix} .$$

Then there is a representative of $S\mathbf{M}$ whose first $k-1$ rows span the rowspace of

$$\begin{pmatrix} \mathbb{1} & \mathbf{v}_2 & \mathbf{0} \end{pmatrix} \mathbf{M} = \begin{pmatrix} \mathbf{A} & \mathbf{b} + \mathbf{v}_2 & \mathbf{0} \end{pmatrix} .$$

Putting this in row reduced echelon form we get

$$\begin{pmatrix} \mathbb{1} & \mathbf{A}^{-1}(\mathbf{b} + \mathbf{v}_2) & \mathbf{0} \end{pmatrix} ,$$

which shows that the second vector in the representation of $S\mathbf{M}$ is equal to $\mathbf{A}^{-1}(\mathbf{b} + \mathbf{v}_2)$.

5.4 Permuting T-values to improve the attack

From Sect. 4.1 we know that the size of $\mathcal{P}(P_N)$ is influenced by the fact that the first two T-values are chosen to be equal to 1. This also impacts the performance of the subgroup chain attack, since at each step we carry out a search in the space of cosets $\mathcal{P}(P_k) \backslash \mathcal{P}(P_k + 1)$. In the first column of Tab. 2 we see that if the T-Values would have been chosen randomly, the most expensive step would have been the first step, where we would have to perform a collision search in a set of at most q^{13} elements. However, Walnut fixes the two first T-values to be 1, so the most expensive step consists of a collision search in a space of at most q^{11} elements. In the last column of Tab. 2 we see that if the designers had chosen to fix the last two T-values to one instead, the complexity of the subgroup chain attack would be reduced: the most expensive step would have been a collision search in a space with only at most q^9 elements. It turns out that we can first apply a transformation to the REM instance to reduce it to an instance of the REM problem where the final two T-values are set to one. Solving this REM instance then only takes $\sqrt{q^9}$ E-Multiplications, so this approach reduces the amount of work by a factor of q. For general values of N, the new method requires approximately $q^{N-7/2}$ E-Multiplications. The reduction relies on the following lemma.

Lemma 4. *Let s_1, s_2 be braids, let (M, π) be a matrix-permutation pair and let T be a set of T-values. Then $s_1 s_2$ is a solution for the REM problem for the pair (M, π) with respect to the list of T-values T if and only if s_2 is a solution for the REM problem for the pair $((CBM(s_1) \downarrow_T)^{-1} M, \sigma(s_1)^{-1}\pi)$ with respect to the permuted list of T-values $\sigma(s_1)(T)$.*

Proof. By applying the definition of E-Multiplication we find that

$$(\mathbb{1}_N, e) \star_T s_1 s_2 = (CBM(s_1) \downarrow_T \cdot \sigma(s_1)(CBM(s_2) \downarrow_T), \sigma(s_1 s_2)) \,.$$

By multiplying from the left by $CBM(s_1) \downarrow_T^{-1}$ and $\sigma(s_1)^{-1}$ we see that the value above is equal to (M, π) if and only if

$$(\sigma(s_1)(CBM(s_2) \downarrow_T), \sigma(b_2)) = ((CBM(s_1) \downarrow_T)^{-1} M, \sigma(s_1)^{-1}\pi) \,.$$

The main insight is that permuting the variables $t_i \mapsto t_{\sigma(b_1)(i)}$ and then evaluating at the values of T leads to the same result as evaluating at the set of permuted values $\sigma(s_1)(T)$. Therefore the left hand side is equal to

$$(CBM(s_2) \downarrow_{\sigma(b_1)(T)}, \sigma(s_2)) = (\mathbb{1}_N, e) \star_{\sigma(s_1)(T)} s_2 \,.$$

Given this lemma, the reduction is straightforward. In order to solve the REM problem for (M, π) we fix a "transport braid" $s_1 = b_2 b_3 \cdots b_{N-1} b_1 b_2 \cdots b_{N-2}$ whose underlying permutation transports the first two entries to the back of the list. Then we calculate the pair $((CBM(s_1) \downarrow_T)^{-1} M, \sigma(s_1)^{-1}\pi)$ and use our REM solving algorithm with respect to the permuted T-values $\sigma(s_1)(T)$ on this pair to find s_2. This is now faster by a factor q because the last two T-values are equal to one. Then $s_1 s_2$ is a solution to the original REM problem.

Table 2. The dimension of the subspaces containing various subgroups, depending on the T-Values

	generic T-values		First two T-values are equal to 1		Last two T-values are equal to 1	
	dim	Δ	dim	Δ	dim	Δ
$\mathcal{P}(P_2)$	1	1	0	0	1	1
$\mathcal{P}(P_3)$	4	3	2	2	4	3
$\mathcal{P}(P_4)$	9	5	5	3	9	5
$\mathcal{P}(P_5)$	16	7	10	5	16	7
$\mathcal{P}(P_6)$	25	9	17	7	25	$\underline{9}$
$\mathcal{P}(P_7)$	36	11	26	9	31	6
$\mathcal{P}(P_8)$	49	$\underline{13}$	37	$\underline{11}$	37	6

5.5 Using a finer chain of subgroups

With a complexity of $O(q^{N/2})$, the factorization algorithm of Hart et al. is more efficient (asymptotically) than the REM solving algorithm that we have described so far. This is due to the fact that Hart et al. use a finer chain of subgroups, which leads to smaller spaces of cosets to search in. In the next paragraph we describe a faster variant of our REM solving algorithm that uses a finer chain of subgroups, similar to the chain used by Hart et al. This variant is much faster than the previous REM solver, but yields solution braids that are longer.

In each step of our REM solving algorithm we have a matrix $M \in A_i$ and we are looking for a braid $s_i \in P_i$ such that $\mathsf{mat}((M,e)\star s_i)$ lies in A_{i-1}. To speed this process up, we can split each step in two substeps. Let C_{i-1} be the subgroup of invertible N-by-N matrices that only differ from the identity matrix in the upper left $(i-1)$-by-$(i-1)$ submatrix. This is a proper subgroup of A_i, which itself contains A_{i-1} as a proper subgroup. To solve the step of the REM solving algorithm we can first search for an $s'_i \in P_i$ such that $\mathsf{mat}((M,e)\star s'_i)$ lies in the intermediate group C_{i-1}, then we search for a braid s''_i such that $\mathsf{mat}((M,e)\star s'_i s''_i)$ lies in A_{i-1}. The first substep of finding s'_i can be carried out with a meet in the middle search. In order to be able to complete the second substep we start by searching for a list of braids $c_1, c_2, \cdots, c_k$ such that $\mathsf{mat}(\mathcal{P}(c_i)) \in C_1$. Then, to solve the second substep, we search for a braid s''_i in the subgroup generated by the braids c_i such that $\mathsf{mat}((M,e)\star s'_i s''_i)$ lies in A_{i-1}.

5.6 Implications and countermeasures

With this method we split each step into two much easier substeps, which greatly improves the efficiency of the algorithm. The downside is that the solutions to the REM problem that are produced are longer than those produced by the original algorithm. This is because the solution now contains braids s''_i which are themselves a concatenation of several slightly longer braids c_i. To avoid inflating the size of the output signature needlessly, it is best to only use this technique for solving the most expensive steps. For 128-bit security parameters the signatures output are longer than legitimately produced signatures, but still small enough to be accepted by the NIST implementation. For 256-bit security parameters, the forged signatures are smaller than some legitimately produced signatures, depending on which variant of the signing algorithm is used. Hence, we cannot defend Walnut against this attack by imposing an upper limit on the length of the signatures. Note that it is trivial to convert a short signature into a longer signature, so imposing a lower bound does not help either.

With this method the most expensive step of the algorithm requires only $q^{N/2-1}$ E-Multiplications. The attack is very efficient in practice. We can produce a forgery for 128-bit security parameters in less than one second. Even for 256-bit security parameters we can forge signatures for any document in less than a minute.

The attack benefits from the fact that two of the T-values are equal to one (see Sect. 5.4), so the attack would be slightly less efficient if the Walnut scheme can be adapted to avoid this. If all T-values are chosen randomly the complexity of the attack becomes dominated by the first step, which requires now roughly $\sqrt{k}q^{(N-1)/2}$ E-multiplications (k, the number of braids produced in the first step is chosen to be 60 in our implementation). Other than this there does not seem to be a better way to block the attack other than just increasing the parameters to ensure that $q^{N/2-1}$ is higher than the desired security level. One way to do this is to take $N = 10, q = 2^{32}$ to achieve 128 bits of security, and $N = 10, q = 2^{64}$ for 256 bits of security.

6 Conclusion

In this paper we presented three different practical methods to break the Walnut digital signature scheme (See Table 3). All three attacks are made possible because of the rich algebraic structure of the E-Multiplication map, which is central to the Walnut scheme (and other protocols developped by SecureRF). The first method exploits a signature malleability property of Walnut, and expands on the work of [14] which attacks an earlier version of the Walnut scheme. The second attack is purely generic. It is much more efficient that expected because E-Multiplication maps a certain subgroup of P_N into a subspace of very low dimension. The last attack exploits the fact that E-Multiplication, when restricted to pure braids, is a homomorphism of groups and that this homomorphism maps the chain of subgroups $P_2 \subset P_3 \subset \cdots \subset P_N$ to a nice chain of subgroups of $GL_N(\mathbf{F}_q)$. Some poor design choices such as adopting an encoding mechanism that produces matrices in a low dimensional subspace and a failed attempt to block the attack of Hart et al. [14] seem to be symptomatic of a lack of understanding of the algebraic structure of E-Multiplication. It is the opinion of the authors that E-Multiplication can not be credibly used as a basis for cryptography until this structure and its implications for cryptography are better understood.

Table 3. An overview of the attacks introduced in this paper, compared with the legitimate signing algorithms.

	Complexity (in number of E-Mults or Mat mults)	128 bits of security		256 bits of security	
		Time	Length of signature (# generators)	Time	Length of signature (# generators)
Signing:					
BKL		< 1 sec	≈ 1480	< 1 sec	≈ 2661
Stochastic		< 1 sec	≈ 2788	< 1 sec	≈ 5260
Attacks:					
factorization	$q^{(N-1)/2}$	5 min	$> 2^{32}$	—	—
collision [4]	$q^{13/2}$	68 min	≈ 1480	—	—
subgroup chain	$q^{N-7/2}$	4 sec	899	58 hours	1374
fine chain	$q^{N/2-1}$	< 1 sec	4534	39 sec	4525

The security of the parameter sets submitted to the NIST PQC project is completely broken by the attacks (see Table 3). It is possible to forge signatures or compute equivalent secret keys in under a second for 128-bit security parameters. Even for 256-bit security parameters this takes less than a minute.

In response to the various attacks, the designers have announced a number of changes to Walnut and increased the parameters (see Table 4) to resist all known attacks. An upper bound of 2^{14} on the number of Artin generators of a signature is imposed, the encoding mechanism is changed so that it outputs braids that map into a larger subspace and the method of producing cloaking elements is changed such that two of the T-values are no longer required to be equal to 1.

Increasing the parameters to resist the attacks introduced in this paper increases the public key by a factor of 10 and the signature sizes by roughly 80%. The updated scheme uses arithmetic in much larger finite fields (e.g. $\mathbb{F}_{2^{61}-1}$ instead of $\mathbb{F}_{2^8}$). This has a relatively small impact on the efficiency of the implementation for high-end processors submitted to NIST (roughly 50% slower signing and 80% slower verification). However, the large finite fields make the scheme more difficult to implement on the low-resource processors that SecureRF is targeting. With the new parameter choices Walnut no longer stands out for its small key and signature sizes relative to other post-quantum signature schemes such as lattice-based, hash-based and multivariate signature schemes. For example, Walnut used to be the signature scheme with the

[4] Has exactly the same length distribution as legitimately produced signatures

[5] Average signature length, computed over 1000 signatures generated with the BKL signing method.

Table 4. Comparison of the original parameter choices with the new parameter that resist the attacks introduced in this paper. Our timing experiments use the implementations that were submitted to NIST, and were run on a Dell OptiPlex 3050 Micro desktop machine.

		Original parameters	New Parameters	Increase
128-bit	N	8	10	
	q	2^5	$2^{31}-1$	
	Public key length	83 Bytes	780 Bytes	×9.4
	Signature length[5]	713 Bytes	1308 Bytes	+83%
	Signing time	39.5 ms	59.2 ms	+50%
	Verification time	0.05 ms	0.09 ms	+80%
256-bit	N	8	10	
	q	2^8	$2^{61}-1$	
	Public key length	128 Bytes	1552 Bytes	×12.1
	Signature length[5]	1296 Bytes	2409 Bytes	+86%
	Signing time	155.2 ms	223.1 ms	+44%
	Verification time	0.07ms	0.20 ms	×2.7

smallest combined size of a public key and a signature out of all the 19 signature schemes submitted to NIST, but this is no longer the case with the new parameters.

Acknowledgements This work was supported in part by the Research Council KU Leuven: C16/15/058. In addition, this work was supported by the European Commission through the Horizon 2020 research and innovation programme under grant agreement No.H2020-ICT-2014-645622 PQCRYPTO. This project was partially supported by the FWO through the WOG Coding Theory and Cryptography. Ward Beullens is funded by an FWO fellowship.

References

1. About SecureRF. https://www.securerf.com/about-us/, accessed: 2018-03-08
2. Anshel, I., Anshel, M., Fisher, B., Goldfeld, D.: New key agreement protocols in braid group cryptography. In: Cryptographers' Track at the RSA Conference. pp. 13–27. Springer (2001)
3. Anshel, I., Anshel, M., Goldfeld, D.: An algebraic method for public-key cryptography. Mathematical Research Letters 6, 287–292 (1999)
4. Anshel, I., Anshel, M., Goldfeld, D., Lemieux, S.: Key agreement, the Algebraic Eraser™, and lightweight cryptography. Contemporary Mathematics 418, 1–34 (2007)

5. Anshel, I., Atkins, D., Goldfeld, D., Gunnells, P.E.: WalnutDSA™: A quantum-resistant digital signature algorithm. IACR eprint 2017/058 (version: 30-Nov-2017)
6. Anshel, I., Atkins, D., Goldfeld, D., Gunnells, P.E.: WalnutDSA™: A quantum-resistant digital signature algorithm. IACR eprint 2017/058 (version: 18-Sept-2017)
7. Anshel, I., Atkins, D., Goldfeld, D., Gunnells, P.E.: A class of hash functions based on the Algebraic Eraser™. Groups Complexity Cryptology 8(1), 1–7 (2016)
8. Anshel, I., Atkins, D., Goldfeld, D., Gunnells, P.E.: The Walnut digital signature algorithm™ specifcation. Submitted to NIST PQC project (2017)
9. Artin, E.: Theory of braids. Annals of Mathematics pp. 101–126 (1947)
10. Ben-Zvi, A., Blackburn, S.R., Tsaban, B.: A practical cryptanalysis of the Algebraic Eraser. In: Advances in Cryptology – CRYPTO 2016. pp. 179–189. Springer (2016)
11. Ben-Zvi, A., Kalka, A., Tsaban, B.: Cryptanalysis via algebraic spans. IACR eprint 41 (2014)
12. Cheon, J.H., Jun, B.: A polynomial time algorithm for the braid Diffie-Hellman conjugacy problem. In: Advances in Cryptology – CRYPTO 2003. pp. 212–225. Springer (2003)
13. Garside, F.A.: The braid group and other groups. The Quarterly Journal of Mathematics 20(1), 235–254 (1969)
14. Hart, D., Kim, D., Micheli, G., Perez, G.P., Petit, C., Quek, Y.: A practical cryptanalysis of WalnutDSA$^{\text{TM}}$ (2018)
15. Hughes, J.: A linear algebraic attack on the AAFG1 braid group cryptosystem. In: Australasian Conference on Information Security and Privacy. pp. 176–189. Springer (2002)
16. Hughes, J., Tannenbaum, A.: Length-based attacks for certain group based encryption rewriting systems. arXiv preprint cs/0306032 (2003)
17. Kalka, A., Teicher, M., Tsaban, B.: Cryptanalysis of the Algebraic Eraser and short expressions of permutations as products. Arxiv preprint (2008)
18. Ko, K.H., Lee, S.J., Cheon, J.H., Han, J.W., Kang, J.s., Park, C.: New public-key cryptosystem using braid groups. In: Advanced in Cryptology – CRYPTO 2000. pp. 166–183. Springer (2000)
19. Myasnikov, A.D., Ushakov, A.: Length based attack and braid groups: cryptanalysis of Anshel-Anshel-Goldfeld key exchange protocol. In: International Workshop on Public Key Cryptography PKC 2007. pp. 76–88. Springer (2007)
20. National Institute for Standards and Technology (NIST): Post-quantum crypto standardization (2016), http://csrc.nist.gov/groups/ST/post-quantum-crypto/
21. Pollard, J.M.: A Monte Carlo method for factorization. BIT Numerical Mathematics 15(3), 331–334 (1975)
22. Sedgewick, R., Szymanski, T.G., Yao, A.C.: The complexity of finding cycles in periodic functions. SIAM Journal on Computing 11(2), 376–390 (1982)
23. Tsaban, B.: Polynomial-time solutions of computational problems in noncommutative-algebraic cryptography. Journal of Cryptology 28(3), 601–622 (2015)
24. Van Oorschot, P.C., Wiener, M.J.: Parallel collision search with cryptanalytic applications. Journal of Cryptology 12(1), 1–28 (1999)

25. Vasco, M.I.G., Steinwandt, R.: A reaction attack on a public key cryptosystem based on the word problem. Applicable Algebra in Engineering, Communication and Computing 14(5), 335–340 (2004)
26. Wagner, N.R., Magyarik, M.R.: A public-key cryptosystem based on the word problem. In: Advances in Cryptology – CRYPTO '84. pp. 19–36. Springer (1984)

Chapter 10

Cryptanalysis of LESS

Know what you deserve. And don't settle for LESS.

– Tony Gaskins

Publication data

Beullens W. Not enough LESS: An improved algorithm for solving Code Equivalence Problems over $\mathbb{F}_q$. In: Selected Areas in Cryptography – SAC 2020. Lecture Notes in Computer Science. Springer, 2020. (Proceedings not yet published at time of writing)

Not enough LESS: An improved algorithm for solving Code Equivalence Problems over $\mathbb{F}_q$

Ward Beullens[1]

imec-COSIC, KU Leuven
ward.beullens@esat.kuleuven.be

Abstract. Recently, a new code based signature scheme, called LESS, was proposed with three concrete instantiations, each aiming to provide 128 bits of classical security [3]. Two instantiations (LESS-I and LESS-II) are based on the conjectured hardness of the linear code equivalence problem, while a third instantiation, LESS-III, is based on the conjectured hardness of the permutation code equivalence problem for weakly self-dual codes. We give an improved algorithm for solving both these problems over sufficiently large finite fields. Our implementation breaks LESS-I and LESS-III in approximately 25 seconds and 2 seconds respectively on an Intel i5-8400H CPU. Since the field size for LESS-II is relatively small ($\mathbb{F}_7$) our algorithm does not improve on existing methods. Nonetheless, we estimate that LESS-II can be broken with approximately 2^{44} row operations.

Keywords: permutation code equivalence problem, linear code equivalence problem, code-based cryptography, post-quantum cryptography

1 Introduction

Two q-ary linear codes $\mathcal{C}_1$ and $\mathcal{C}_2$ of length n and dimension k are called permutation equivalent if there exists a permutation $\pi \in S_n$ such that $\pi(\mathcal{C}_1) = \mathcal{C}_2$. Similarly, if there exists a monomial permutation $\mu \in M_n = (\mathbb{F}_q^\times)^n \ltimes S_n$ such

This work was supported by CyberSecurity Research Flanders with reference number VR20192203 and the Research Council KU Leuven grants C14/18/067 and STG/17/019. Ward Beullens is funded by FWO SB fellowship 1S95620N.

that $\mu(\mathcal{C}_1) = \mathcal{C}_2$ the codes are said to be linearly equivalent (a monomial permutation acts on vectors in $\mathbb{F}_q^n$ by permuting the entries and also multiplying each entry with a unit of $\mathbb{F}_q$). The problem of finding $\pi \in S_n$ (or $\mu \in M_n$) given equivalent $\mathcal{C}_1$ and $\mathcal{C}_2$ is called the permutation equivalence problem (or linear equivalence problem respectively)[1].

Definition 1 (Permutation Code Equivalence Problem). *Given generator matrices of two permutation equivalent codes $\mathcal{C}_1$ and $\mathcal{C}_2$, find a permutation $\pi \in S_n$ such that $\mathcal{C}_2 = \pi(\mathcal{C}_1)$.*

Definition 2 (Linear Code Equivalence Problem). *Given generator matrices of two linearly equivalent codes $\mathcal{C}_1$ and $\mathcal{C}_2$, find a monomial permutation $\mu \in M_n$ such that $\mathcal{C}_2 = \mu(\mathcal{C}_1)$.*

The hardness of the permutation equivalence problem is relevant for the security of the McEliece and Girault post-quantum cryptosystems [10,7]. More recently, Biasse, Micheli, Persichetti, and Santini proposed a new code-based signature scheme whose security only relies on the hardness of the linear code equivalence problem or permutation code equivalence problem. The public key consists of generator matrices for two equivalent codes $\mathcal{C}_1$ and $\mathcal{C}_2$, and a signature is a zero-knowledge proof of knowledge of an equivalence $\mu \in M_n$ (or $\pi \in S_n$) such that $\mu(\mathcal{C}_1) = \mathcal{C}_2$ (or $\pi(\mathcal{C}_1) = \mathcal{C}_2$ respectively). In the case of permutation equivalence, the codes $\mathcal{C}_1$ and $\mathcal{C}_2$ are chosen to be weakly self-dual, because otherwise π can be recovered in polynomial time [12]. Table 1 shows the proposed parameter sets for LESS, aiming for 128 bits of security.

Parameter set	n	k	p	equivalence
LESS-I	54	27	53	Linear
LESS-II	106	45	7	Linear
LESS-III	60	25	31	Permutation

Table 1. Proposed parameter sets for the LESS signature scheme.

[1] There also exists a more general notion of equivalence called semi-linear equivalence. Our methods generalize to semi-linear equivalences, but since this is not relevant for the security of LESS, we do not elaborate on this.

1.1 Previous work

We will briefly go over some of the algorithms that have been proposed for the permutation and linear code equivalence problems below. The state of the art for the permutation code equivalence problem is that random instances can be solved in polynomial time with the Support Splitting Algorithm (SSA), but that instances with codes that have large hulls require a runtime that is exponential in the dimension of the hull. Weakly self-dual codes (these are codes $\mathcal{C}$ such that $\mathcal{C} \subset \mathcal{C}^{\perp}$) have hulls of maximal dimension $\dim(\mathcal{H}(\mathcal{C})) = \dim(\mathcal{C})$ and are believed to be the hardest instances of the permutation equivalence problem. The state of the art for the linear code equivalence problem is that instances over $\mathbb{F}_q$ with $q \leq 4$ can be solved in polynomial time with the SSA algorithm via a reduction to the permutation equivalence problem, but for $q > 4$ this reduction results in codes with a large hull, which means the SSA algorithm is not efficient. Hence, the linear code equivalence problem is conjectured to be hard on average for $q > 4$ [13].

Leon's Algorithm. Leon's algorithm [9] for finding linear and permutation equivalences relies on the observation that applying a permutation or a monomial permutation does not change the hamming weight of a codeword. Therefore, if we compute the sets X_1 and X_2 of all the minimal-weight codewords of $\mathcal{C}_1$ and $\mathcal{C}_2$ respectively, then it must be that $X_2 = \pi(X_1)$ or $X_2 = \mu(X_1)$ in the case of permutation equivalence or linear equivalence respectively. Leon gives an algorithm to compute a $\mu \in M_n$ that satisfies $X_2 = \mu(X_1)$ with a time complexity that is polynomial in $|X_1|$. Usually, the sets X_1 and X_2 have "enough structure", such that if μ satisfies $X_2 = \mu(X_1)$, then also $\mathcal{C}_2 = \mu(\mathcal{C}_1)$ with non-negligible probability. If this is not the case, then one can also consider larger sets X_1' and X_2' that contain all the codewords in $\mathcal{C}_1$ and $\mathcal{C}_2$ respectively whose weight is one more than the minimal weight. Since the sets X_1 and X_2 are usually small, the complexity of the algorithm is dominated by the complexity of computing X_1 and X_2.

Feulner gives an algorithm that computes a canonical representative of an equivalence class of codes. The complexity of this algorithm is close to that of Leon's algorithm [6].

Support Splitting Algorithm. The Support Splitting Slgorithm (SSA) of Sendrier [12] defines the concept of a signature. A signature is a property of a position in a code that is invariant for permutations. More precicely, it is a function $\mathcal{S}$ that takes a code $\mathcal{C}$ and a position $i \in \{1, \cdots, n\}$ as input and outputs an element of an output space P, such that for any permutation $\pi \in S_n$ we have

$$\mathcal{S}(\mathcal{C}, i) = \mathcal{S}(\pi(\mathcal{C}), \pi(i))\,.$$

We say that a signature is totally discriminant for $\mathcal{C}$ if $i \neq j$ implies that $\mathcal{S}(\mathcal{C}, i) \neq \mathcal{S}(\mathcal{C}, j)$. If a signature $\mathcal{S}$ is efficiently computable and totally discriminant for a code $\mathcal{C}_1$, then one can easily solve the permutation equivalence problem by computing $\mathcal{S}(\mathcal{C}_1, i)$ and $\mathcal{S}(\mathcal{C}_2, i)$ for all $i \in \{1, \cdots, n\}$ and comparing the outputs. Even if the signature is not totally discriminant, a sufficiently discriminant signature can still be used to solve the permutation equivalence Problem by iteratively refining the signature.

The SSA uses the concept of the hull of a code to construct an efficiently computable signature. The hull of a code $\mathcal{C}$ is the intersection of the code with its dual: $\mathcal{H}(\mathcal{C}) = \mathcal{C} \cap \mathcal{C}^{\perp}$. This concept is very useful in the context of the permutation equivalence problem because taking the hull commutes with applying a permutation[2]

$$\mathcal{H}(\pi(\mathcal{C})) = \pi(\mathcal{C}) \cap \pi(\mathcal{C})^{\perp} = \pi(\mathcal{C} \cap \mathcal{C}^{\perp}) = \pi(\mathcal{H}(\mathcal{C}))\,.$$

The SSA defines a signature as $\mathcal{S}(C, i) := W(\mathcal{H}(C_i))$, where $\mathcal{C}_i$ is the code $\mathcal{C}$ punctured at position i, and $W(\mathcal{C})$ denotes the weight enumerator polynomial of the code $\mathcal{C}$. While this signature is typically not fully discriminant, it is still discriminant enough to efficiently solve the permutation equivalence Problem for random matrices. However, a limitation of the SSA algorithm is that computing the enumerator of the hull is not efficient when the hull of $\mathcal{C}$ is large. For random codes this is not a problem because typically the hull is small.

Algebraic approach. The code equivalence problems can be solved algebraically, by expressing the condition $\pi(\mathcal{C}_1) = \mathcal{C}_2$ or $\mu(\mathcal{C}_1) = \mathcal{C}_2$ as a system of polynomial equations, and trying to solve this system with Gröbner basis methods [11]. Similar to the SSA algorithm, this solves the permutation code equivalence problem for random instances in polynomial time, but the complexity is exponential in the dimension of the hull. The approach also works for the linear code equivalence problem, but it is only efficient for $q \leq 4$.

[2] This is not the case for monomial permutations: $\mathcal{H}(\mu(\mathcal{C}))$ is not necessarily equal to $\mu(\mathcal{H}(\mathcal{C}))$ for $\mu \in M_n$. This is why the SSA can not be directly applied to find linear equivalences.

1.2 Our contributions

In this paper, we propose an improvement on Leon's algorithm for code equivalence that works best over sufficiently large finite fields. If $\mathbf{x} \in \mathcal{C}_1$ and $\mathbf{y} = \pi(\mathbf{x}) \in \mathcal{C}_2$, then the multiset of entries of $\mathbf{x}$ matches the multiset of entries of $\mathbf{y}$. Our algorithm is based on the observation that if the size of the finite field is large enough then the implication also holds in the other direction with large probability: If $\mathbf{x} \in \mathcal{C}_1$ and $\mathbf{y} \in \mathcal{C}_2$ are low-weight codewords with the same multiset of entries, then with large probability $\pi(\mathbf{x}) = \mathbf{y}$. Our algorithm does a collision search to find a small number of such pairs $(\mathbf{x}, \mathbf{y} = \pi(\mathbf{x}))$, from which one can easily recover π. We also give a generalization of this idea that works for the linear equivalence problem.

We implemented our algorithm and used it to break the LESS signature scheme. In the LESS-I and LESS-III parameter sets the finite field is large enough for our algorithm to improve on Leons's algorithm. We show that we can recover a LESS-I or LESS-III secret key in only 25 seconds or 2 seconds respectively. We estimate that recovering the secret key is also possible in practice with Leon's algorithm, but it would be significantly more costly. LESS-II works over $\mathbb{F}_7$, which is too small for our algorithm to improve on Leon's algorithm: We estimate that our algorithm requires approximately $2^{50.4}$ row operations, while Leon's algorithm would take only $2^{43.9}$ row operations.

2 Preliminaries

2.1 Notation.

For a q-ary linear code $\mathcal{C}$ of length n and dimension k we say a matrix $\mathbf{G} \in \mathbb{F}_q^{k \times n}$ is a generator matrix for $\mathcal{C}$ if $\mathcal{C} = \langle \mathbf{G} \rangle$, where $\langle \mathbf{G} \rangle$ denotes the span of the rows of $\mathbf{G}$. Similarly, we say that a matrix $\mathbf{H} \in \mathbb{F}_q^{(n-k) \times n}$ is a parity check matrix for $\mathcal{C}$ if $\mathcal{C}^\perp = \langle H \rangle$, where $\mathcal{C}^\perp = \{\mathbf{x} \in \mathbb{F}_q^n \,|\, \mathbf{x} \cdot \mathbf{y} = 0 \, \forall \mathbf{y} \in \mathcal{C}\}$ is the dual code of $\mathcal{C}$. For a vector $\mathbf{x} \in \mathbb{F}_q^n$ we denote by $\mathrm{Supp}(\mathbf{x})$ the set of indices of the nonzero entries in $\mathbf{x}$, i.e. $\mathrm{Supp}(\mathbf{x}) = \{i \,|\, x_i \neq 0\}$. We define the support $\mathrm{Supp}(\mathcal{C})$ of a code $\mathcal{C}$ to be the union of the supports of the codewords in $\mathcal{C}$. We let $wt(\mathbf{x}) = |\mathrm{Supp}(\mathbf{x})|$ be the *Hamming weight* of $\mathbf{x}$. We denote by $B_n(w)$ the *Hamming ball* with radius w, i.e. the set of vectors in $\mathbf{x} \in \mathbb{F}_q^n$ with $wt(x) \leq w$. For a permutation $\pi \in S_n$ and a vector $\mathbf{x}$ of length n, we write $\pi(\mathbf{x})$ for the vector obtained by permuting the entries of $\mathbf{x}$ with the permutation π, that is we have $(\pi(\mathbf{x}))_i = \mathbf{x}_{\pi(i)}$ for all

$i \in \{1, \cdots, n\}$. For a monomial permutation $\mu = (\nu, \pi) \in M_n = (\mathbb{F}_q^\times)^n \ltimes S_n$ and a vector $\mathbf{x} \in \mathbb{F}_q^n$, we write $\mu(\mathbf{x})$ to denote the vector obtained by applying μ to the entries of $\mathbf{x}$. Concretely, we have $(\mu(x))_i = \nu_i \cdot \mathbf{x}_{\pi(i)}$ for all $i \in \{1, \cdots, n\}$. For a code $\mathcal{C}$ and $\pi \in S_n$ (or $\mu \in M_n$), we denote by $\pi(\mathcal{C})$ (or $\mu(\mathcal{C})$) the code that consist of permutations (or monomial permutations respectively) of codewords in $\mathcal{C}$.

2.2 Information set decoding.

The algorithms in this paper will make use of information set decoding to find sparse vectors in q-ary linear codes. In particular, we will use the Lee-Brickell algorithm with parameter $p = 2$. To find low-weight codewords in a code $\mathcal{C} = \langle \mathbf{M} \rangle$ the algorithm repeatedly computes the echelon form of M with respect to a random choice of k pivot columns. Then, the algorithm inspects all the linear combinations of $p = 2$ rows of the matrix. Given the echelon form of the matrix, we are guaranteed that all these linear combinations have weight at most $n - k + 2$, but if we are lucky enough we will find codewords that are even more sparse. We repeat this until a sufficiently sparse codeword is found.

Complexity of the algorithm. The complexity of the algorithm depends on the length n and the dimension k of the code, target weight w, and whether we want to find a single codeword, all the codewords, or a large number N of codewords. First, suppose there is a distinguished codeword $\mathbf{x} \in \mathcal{C}$ with weight w that we want to find. For a random choice of pivot columns, the Lee-Brickell algorithm will output $\mathbf{x}$ if the support of $\mathbf{x}$ intersects the set of pivot columns (also known as the information set) in exactly 2 positions. The probability that this happens is

$$P_\infty(n, k, w) := \frac{\binom{n-k}{w-2}\binom{k}{2}}{\binom{n}{w}} .$$

Therefore, since the cost of each iteration is k^2 row operations for the Gaussian elimination and $\binom{k}{2}q$ row operations to iterate over all the linear combinations of 2 rows (up to multiplication by a constant), the algorithm will find $\mathbf{x}$ after approximately

$$C_\infty(n, k, w) = \left(k^2 + \binom{k}{2}q\right) P(n, k, w)^{-1} = O\left(\frac{q\binom{n}{w}}{\binom{n-k}{w-2}}\right)$$

row operations. Heuristically, for random codes we expect the support of the different codewords to behave as if they are "independent", so if there exist

$(q-1)N$ codewords of weight w (i.e. N different codewords up to multiplication by a scalar), then we expect the probability that one iteration of the Lee-Brickell algorithm succeeds to be

$$P_1(n,k,w) = 1 - (1 - P_\infty(n,k,w))^N .$$

Thus, if N is small enough, we have $P_1(n,k,w) \approx NP_\infty(n,k,w)$, and the complexity of finding is a single weight-w codeword is $C_1(n,k,w) \approx C_\infty(n,k,w)/N$.

Finally, if the goal is to find L out of the N distinct weight-w codewords (up to multiplication by a scalar), the cost of finding the first codeword is $C_1 = C_\infty(n,k,w)/N$, the cost of finding the second codeword is $C_\infty(n,k,w)/(N-1)$, the cost of finding the third codeword is $C_\infty(n,k,w)/(N-2)$ and so on. Summing up these costs, we get that the cost of finding L distinct codewords is

$$C_L(n,k,w) \approx C_\infty(n,k,w) \cdot \left(\sum_{i=0}^{L-1} \frac{1}{N-i} \right) .$$

Therefore, if $L << N$, we can estimate $C_L \approx C_\infty L/N$, and if the goal is to find all the codewords, we get $C_N \approx C_\infty \ln(N)$, where ln denotes the natural logarithm, because $\sum_{i=1}^{N} 1/i \approx \ln(N)$.

3 New algorithm for Permutation Equivalences over $\mathbb{F}_q$

In this section, we introduce an algorithm for the permutation equivalence Problem over sufficiently large fields $\mathbb{F}_q$ (which is the case of the LESS-III parameter set). The complexity of the algorithm is independent of the size of the hull of the equivalent codes. Therefore, our algorithm can be used to find equivalences when the hull is so large that using the SSA algorithm becomes infeasible. The complexity of the algorithm is better than Leon's algorithm when the size of the finite field is sufficiently large.

Main idea. Leon's algorithm computes the sets $X_1 = \mathcal{C}_1 \cap B_n(w_{min})$ and $X_2 = \mathcal{C}_2 \cap B_n(w_{min})$, where w_{min} is the minimal weight of codewords in $\mathcal{C}_1$ and solves the Code equivalence problem by looking for $\pi \in S_n$ such that $\pi(X_1) = X_2$. The bottleneck of Leon's algorithm is computing $X_1 = \mathcal{C}_1 \cap B_n(w_{min})$ and $X_2 = \mathcal{C}_2 \cap B_n(w_{min})$, so if we want to improve the complexity of the attack we need to avoid computing all of X_1 and X_2. An easy observation is that permuting a codeword $\mathbf{x}$ does not only preserve its Hamming weight, but also the multiset of entries of $\mathbf{x}$. Therefore, if there is an element $\mathbf{x} \in X_1$ with a unique multiset, then one can immediately see to which vector $\mathbf{y} = \pi(\mathbf{x}) \in X_2$ it gets mapped. For example, if X_1 contains the vector

$$\mathbf{x} = \begin{pmatrix} 0 & 4 & 0 & 0 & 7 & 4 & 0 & 0 & 0 & 14 \end{pmatrix} ,$$

and if X_2 contains the vector

$$\mathbf{y} = \begin{pmatrix} 0 & 4 & 7 & 0 & 0 & 14 & 0 & 4 & 0 & 0 \end{pmatrix} ,$$

then assuming there are no other vectors in X_1 and X_2 with the same multiset of entries, we know that $\pi(\mathbf{x}) = \mathbf{y}$. In particular we know that $\pi(5) = 3$, $\pi(10) = 6$ and $\pi(2), \pi(6) \in \{2, 8\}$.

Instead of computing all of X_1 and X_2, our algorithm will search for a small number of such pairs $(\mathbf{x}, \mathbf{y} = \pi(\mathbf{x}))$, which will give enough information to determine π. This approach will not work if $\mathbb{F}_q$ is too small, because then X_1 will contain a lot of vectors with the same multiset of entries. (E.g. in the case $q = 2$, all the vectors with the same weight have the same multiset of entries.)

If we compute only $\Theta(\sqrt{|X_1| \log n})$ elements of X_1 and X_2, then we expect to find $\Theta(\log n)$ pairs $(\mathbf{x}, \mathbf{y} = \pi(\mathbf{x}))$, which suffices to recover π. This speeds up the procedure by a factor $\Theta(\sqrt{|X_1| / \log n})$, which is only a small factor. We can improve this further by considering larger sets $X_1' = \mathcal{C}_1 \cap B_n(w)$ and $X_2' = \mathcal{C}_2 \cap B_n(w)$ for a weight w that is not minimal. In the most favorable case where the multisets of the vectors in X_i' are still unique for $w = n - k + 1$, then we can sample from X_1' and X_2' in polynomial time using Gaussian elimination, and we get an algorithm that runs in time $\tilde{O}\left(\sqrt{\binom{n}{k-1}}\right)$, where $\tilde{O}$ is the usual big-O notation but ignoring polynomial factors.

Description of the algorithm. The algorithm works as follows:

1. Let w be maximal subject to $\frac{n!}{(n-w)!} q^{-n+k} < \frac{1}{4 \log n}$ and $w \leq n - k + 1$.
2. Repeatedly use information set decoding to generate a list L that contains $\sqrt{|B_n(w)| \cdot q^{-n+k-1} \cdot 2 \log n}$ pairs of the form $(\mathbf{x}, \mathsf{lex}(\mathbf{x}))$, where $\mathbf{x} \in \mathcal{C}_1 \cap B_n(w)$ and where $\mathsf{lex}(\mathbf{x})$ is the lexicographically first element of the set $\{\pi(\alpha \mathbf{x}) | \pi \in S_n, \alpha \in \mathbb{F}_q^\times\}$.

3. Initialize an empty list P and repeatedly use information set decoding to generate $\mathbf{y} \in \mathcal{C}_2 \cap B_n(w)$. If there is a pair $(\mathbf{x}, \mathsf{lex}(\mathbf{x}))$ in L such that $\mathsf{lex}(\mathbf{x}) = \mathsf{lex}(\mathbf{y})$, then append $(\mathbf{x}, \mathbf{y})$ to P. Continue until P has $2\log(n)$ elements.

4. Use a backtracking algorithm to iterate over all permutations π that satisfy $\langle \pi(\mathbf{x}) \rangle = \langle \mathbf{y} \rangle$ for all $(\mathbf{x}, \mathbf{y}) \in P$ until a permutation is found that satisfies $\pi(C_1) = C_2$.

Heuristic analysis of the algorithm. Heuristically, we expect that for $\mathbf{x} \in \mathcal{C}_1 \cap B_n(w)$ the probability that there exists $\mathbf{x}' \in \mathcal{C}_1 \cap B_n(w)$ such that $\langle \mathbf{x}' \rangle \neq \langle \mathbf{x} \rangle$ and $\mathsf{lex}(\mathbf{x}) = \mathsf{lex}(\mathbf{x}')$ to be bounded by $\frac{n!}{(n-w)!} q^{-n+k}$, because there are at most $\frac{n!}{(n-w)!}$ values of $\mathbf{x}'$ (up to multiplication by a unit) for which $\mathsf{lex}(\mathbf{x}') = \mathsf{lex}(\mathbf{x})$, (namely all the permutations $\mathbf{x}$), and each of these vectors is expected to be in $\mathcal{C}_1$ with probability $q^{-(n-k)}$. In step 1 of the algorithm we choose w such that the probability estimate that $\mathbf{x}$ is part of such a collision in lex is at most $1/(4\log n)$.

We have $\mathsf{lex}(\mathcal{C}_1 \cap B_n(w)) = \mathsf{lex}(\mathcal{C}_2 \cap B_n(w))$ and heuristically the size of this set is close to $|\mathcal{C}_1 \cap B_n(w)|/(q-1) \approx |B_n(w)|/q^{n-k+1}$ since lex is almost $(q-1)$-to-one. Therefore, it takes roughly $|B_n(w)| 2\log n / q^{n-k+1} |L|$ iterations of step 3 until $2 \log n$ pairs $(\mathbf{x}, \mathbf{y})$ with $\mathsf{lex}(\mathbf{x}) = \mathsf{lex}(y)$ are found. We chose the list size $|L| = \sqrt{|B_n(w)| 2\log n / q^{n-k+1}}$ so that the work in step 2 and step 3 is balanced.

The last part of the algorithm assumes that for each pair $(\mathbf{x}, \mathbf{y})$ found in step 3 we have $\langle \pi(\mathbf{x}) \rangle = \langle \mathbf{y} \rangle$. This can only fail with probability bounded by $1/4\log n$, because this implies that $\pi(\mathbf{x})$ and $\mathbf{y} \in \mathcal{C}_2 \cap B_n(w)$ form a collision for lex. Summing over all the $2\log n$ pairs we get that the probability that $\langle \pi(\mathbf{x}) \rangle = \langle \mathbf{y} \rangle$ holds for all the pairs in P is at least $1/2$. If this is the case then there are typically very few permutations σ (most of the time only one) that satisfy $\langle \sigma(\mathbf{x}) \rangle = \langle \mathbf{y} \rangle$ and the true code equivalence π must be one of them.

The complexity of the attack is dominated by the cost of the ISD algorithm to find $|L|$ weight-w codewords in $\mathcal{C}_1$ and $\mathcal{C}_2$ in step 2 and 3, which is

$$2 \cdot C_{|L|}(n, k, w)$$

In our implementation we have used the Lee-Brickell algrithm [8] with $p = 2$ to instantiate the ISD oracle[3]. In this case, the number of row-operations used by the ISD algorithm can be approximated (see section 2.2) as

$$2 \cdot C_{|L|}(n,k,w) \approx C_\infty \frac{|L|}{|\mathcal{C}_1 \cap B_n(w)|/(q-1)} = O\left(\frac{\binom{n}{w}\sqrt{\log n}}{\binom{n-k}{w-2} \cdot \sqrt{|B_n(w)|q^{-n+k}}} \right) .$$

The algorithm in practice. An implementation of our algorithm in C, as well as a python script to estimate the complexity of our attack is made publicly available at

`www.github.com/WardBeullens/LESS_Attack`.

We used this implementation to break the LESS-III parameter set. The public key of a LESS-III signature consist of two permutation equivalent codes $\mathcal{C}_1$ and $\mathcal{C}_2$ of length $n = 60$ and dimension $k = 25$ over $\mathbb{F}_{31}$. The codes are chosen to be weakly self-dual. From experiments, we see that the weakly self-dual property does not seem to affect the complexity or the success rate of our attack.

For these parameters, the maximal value of w that satisfies

$$\frac{n!}{(n-w)!} q^{-n+k} < \frac{1}{4 \log n}$$

is $w = 30$, so we use the Lee-Brickell algorithm to find codewords in $\mathcal{C}_1$ and $\mathcal{C}_2$ with Hamming weight at most 30. The list size is $\sqrt{|B_n(w)| \cdot q^{-n+k-1} \cdot 2 \log n}$, which is close to 25000. With these parameter choices, the algorithm runs in about 2 seconds on a laptop with an Intel i5-8400H CPU at 2.50GHz. The rate at which pairs are found closely matched the heuristic analysis of the previous section: The analysis suggests that we should have to do approximately $2^{14.7}$ Gaussian eliminations, while the average number of Gaussian eliminations measured in our experiments is $2^{14.6}$. However, we find that the heuristic lower bound of 1/2 for the success probability is not tight: The algorithm terminates successfully in all of the executions. This is because in our heuristic analysis we used $n!/(n-w)!$ as an upper bound for the number of permutations of a vector $\mathbf{x}$ of weight w. This upper bound is only achieved if all the entries of $\mathbf{x}$ are distinct. For a random vector $\mathbf{x}$ the number of permutations is much smaller,

[3] One can also use more advanced ISD algorithms such as Stern's algorithm [14], but since we will be working with relatively large finite fields we found that this does not offer a big speedup. To simplify the analysis and the implementation we have chosen for the Lee-Brickell algorithm.

which explains why the observed probability of a bad collision is much lower than our heuristic upper bound.

Remark 1. If we use the algorithm for longer codes the list L will quickly be so large that it would be very costly to store the entire list in memory. To avoid this we can define 2 functions F_1 and F_2 that take a random seed as input, run an ISD algorithm to find a weight w codeword $\mathbf{x}$ in $\mathcal{C}_1$ or $\mathcal{C}_2$ respectively and output $\mathsf{lex}(\mathbf{x})$. Then we can use a memoryless claw-finding algorithm such as the Van Oorschot-Wiener algorithm [16] to find inputs a, b such that $F_1(a) = F_2(b)$. This makes the memory complexity of the algorithm polynomial, at essentially no cost in time complexity. Since memory is not an issue for attacking the LESS parameters we did not implement this approach.

Comparison with Leon's algorithm and new parameters for LESS. We expect recovering a LESS-III secret key with Leon's algorithm would require $2^{24.5}$ iterations of the Lee-Brickell algorithm, significantly more than the $2^{14.6}$ iterations that our algorithm requires. Figure 1 shows how the complexity of our attack and Leon's attack scales with increasing code length n. The left graph shows the situation where the field size q and the dimension k increases linearly with the code length, while the graph on the right shows the case where $q = 31$ is fixed. In both cases, our algorithm outperforms Leon's algorithm, but since our algorithm can exploit the large field size, the gap is larger in the first case. The sawtooth-like behavior of the complexity of Leon's algorithm is related to the number of vectors of minimal weight, which oscillates up and down. We see that in order to achieve 128 bits of security (i.e. an attack needs 2^{128} row operations) we can use a q-ary code of length $n = 280$, dimension $k = 117$ and $q = 149$. Alternatively, if we keep $q = 31$ fixed, we could use a code of length $n = 305$ and dimension $k = 127$. This would result in an average signature size of 18.8 KB or 21.1 KB respectively. This is almost a factor 5 larger than the current signature size of 3.8 KB [4]. The public key size would increase from 0.53 KB [5] to 16.8 KB or 13.8 KB for the $q = 149$ or $q = 31$ parameter set respectively, an increase of more than a factor 25. The fact that our algorithm performs better in comparison to Leon's algorithm for larger finite fields is illustrated in fig. 2, where we plot the complexity of both algorithms for $n = 250$, $k = 104$ and for various field sizes.

[4] The LESS paper claims 7.8 KB. but 4 KB of the signature consists of commitments that can be recomputed by the verifier, so this does not need to be included in the signature size.

[5] The LESS paper claims 0.9 KB public keys, but the generator matrix can be put in normal form, which reduces the size from $k \times n$ field elements to $k \times (n-k)$ field elements.

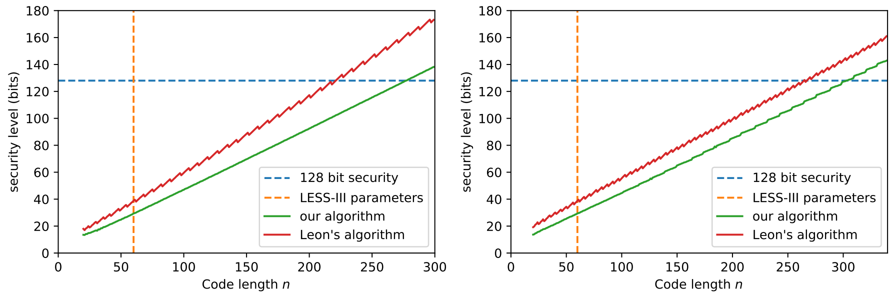


Fig. 1. Complexity of Leon's algorithm and our algorithm for finding permutation equivalences in function of the code Length. In the left graph the field size scales linearly with the code length, in the right graph the field size $q = 31$ is fixed. In both cases the rate of the code is fixed at $k/n = 5/12$.

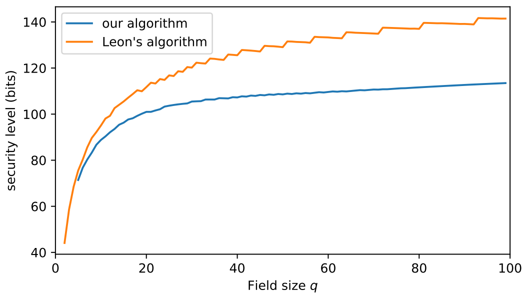


Fig. 2. Complexity of Leon's algorithm and our algorithm for finding permutation equivalences in function of the finite field size for random linear codes of length $n = 250$ and dimension $k = 104$.

4 New algorithm for Linear Equivalences over $\mathbb{F}_q$

In this section, we generalize the algorithm from the previous section to the linear equivalence Problem. The main obstacle we need to overcome is that it does not seem possible given sparse vectors $\mathbf{x} \in \mathcal{C}_1$ and $\mathbf{y} \in \mathcal{C}_2$ to verify if $\mu(\mathbf{x}) = \mathbf{y}$, where $\mu \in M_n$ is the monomial transformation such that $\mu(\mathcal{C}_1) = \mathcal{C}_2$. In the permutation equivalence setting, we could guess that if the multiset of entries of

$\mathbf{x}$ equals the multiset of entries of $\mathbf{y}$ then $\pi(\mathbf{x}) = \mathbf{y}$. If the size of the finite field was large enough, then this was correct with large probability. This strategy does not work in the linear equivalence setting, because monomial permutations do not preserve the multiset of entries. In fact, monomial transformations do not preserve anything beyond the hamming weight of a vector, because for any two codewords $\mathbf{x}$ and $\mathbf{y}$ with the same weight there exists $\mu \in M_n$ such that $\mu(\mathbf{x}) = \mathbf{y}$.

Main Idea. To overcome this problem, be will replace sparse vectors by 2-dimensional subspaces with small support. Let

$$X_1(w) = \{V \subset \mathcal{C}_1 | \dim(V) = 2 \text{ and } |\mathrm{Supp}(V)| \leq w\}$$

be the set of 2 dimensional linear subspaces of $\mathcal{C}_1$ with support of size at most w and similarly we let $X_2(w)$ be the set of 2-spaces in $\mathcal{C}_2$ with support of size w. If $\mu \in M_n$ is a monomial permutation such that $\mu(\mathcal{C}_1) = \mathcal{C}_2$, then for all $V \in X_1(w)$ we have $\mu(V) \in X_2$. Analogously with the algorithm from the previous section, we will sample 2-spaces from $X_1(w)$ and from $X_2(w)$ in the hope of finding spaces $V \in X_1(w)$ and $U \in X_2(w)$ such that $\mu(V) = W$. Then, after finding $\Omega(\log(n))$ such pairs we expect to be able to recover the equivalence μ. To detect if $\mu(V) = W$ we define $\mathsf{lex}(V)$ to be the lexicographically first basis of a 2-space in the M_n-orbit of V. Clearly, if $\mu(V) = W$, then the M_n-orbits of V and W will be the same and hence $\mathsf{lex}(V) = \mathsf{lex}(W)$. Moreover, since the dimension of V and W is only 2, it is feasible to compute $\mathsf{lex}(V)$ and $\mathsf{lex}(W)$ efficiently.

Computing lex(V). To compute $\mathsf{lex}(V)$ we can simply consider all the bases $\mathbf{x}, \mathbf{y}$ that generate V (there are $(q^2-1)(q^2-q)$ of them) and for each of them find the monomial transformation μ such that $\mu(\mathbf{x}), \mu(\mathbf{y})$ comes first lexicographically, and then take the permuted basis that comes first out of these $(q^2-1)(q^2-q)$ options. Given a basis $\mathbf{x}, \mathbf{y}$, finding the lexicographically first value of $\mu(\mathbf{x}), \mu(\mathbf{y})$ is relatively straightforward: First make sure that $\mu(\mathbf{x})$ is minimal, and then use the remaining degrees of freedom to minimize $\mu(\mathbf{y})$. The minimal $\mu(\mathbf{x})$ consists of $n - wt(\mathbf{x})$ zeroes followed by $wt(\mathbf{x})$ ones, which is achieved by multiplying the non-zero entries of $\mathbf{x}$ (and the corresponding entries of $\mathbf{y}$) by their inverse and permuting $\mathbf{x}$ such that all the ones are in the back. The remaining degrees of freedom of μ can be used to make the first $n - wt((\mathbf{x})$ entries of $\mu(\mathbf{y})$ consist of a number of zeros followed by a number of ones and to sort the remaining entries of $\mu(y)$ in ascending order.

A basis $\mathbf{x}, \mathbf{y}$ for V can only lead to the lexicographicallt first $\mu(\mathbf{x}), \mu(\mathbf{y})$ if the hamming weight of $\mathbf{x}$ is minimal among all the vectors in V. Therefore, we only need to consider bases $\mathbf{x}, \mathbf{y}$ where the hamming weight of $\mathbf{x}$ is minimal. When the first basis vector is fixed, choosing the second basis vector and minimizing the basis, can be on average done with a constant number row operations, so the average cost of the algorithm is $q+1+O(N) = O(q)$ row operations, where the $q+1$ operations stem from finding the minimal weigth vectors in V, and N is the number of such vectors.

Example 1. The following is an example what $\mathsf{lex}(V)$ could look like:

$$V = \left\langle \begin{pmatrix} 19 & 3 & 21 & 36 & 17 & 44 & 0 & 47 & 34 & 19 & 48 & 3 & 0 & 47 & 0 & 38 & 27 & 8 & 49 & 18 & 8 & 0 & 0 & 31 & 26 & 52 & 7 & 30 & 37 & 47 \\ 35 & 24 & 13 & 0 & 50 & 40 & 0 & 52 & 6 & 19 & 37 & 28 & 0 & 13 & 0 & 49 & 34 & 20 & 24 & 30 & 24 & 45 & 0 & 39 & 42 & 0 & 18 & 17 & 28 & 36 \end{pmatrix} \right\rangle$$

$$\mathsf{lex}(V) = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 2 & 3 & 4 & 5 & 9 & 9 & 11 & 13 & 15 & 19 & 20 & 21 & 23 & 27 & 32 & 33 & 34 & 36 & 37 & 39 \end{pmatrix}$$

Description of the algorithm. The algorithm works as follows:

1. Let w be maximal subject to $\frac{n!}{(n-w)!}q^{w-1+2k-2n} < \frac{1}{4\log n}$ and $w \leq n-k+2$.
2. Repeatedly use information set decoding to generate a lists L that contains $\sqrt{\binom{n}{w} \cdot q^{2(-n+k+w-2)} \cdot 2\log n}$ pairs of the form $(V, \mathsf{lex}(V))$, where $V \in X_1(w)$.
3. Initialize an empty list P and repeatedly use information set decoding to generate $W \in X_2$. If there is a pair $(V, \mathsf{lex}(V))$ in L such that $\mathsf{lex}(V) = \mathsf{lex}(W)$, then append (V, W) to P. Continue until P has $2\log(n)$ elements.
4. Use a backtracking algorithm to iterate over all monomial permutations μ that satisfy $\mu(V) = W$ for all $(V, W) \in P$ until a monomial permutation is found that satisfies $\mu(\mathcal{C}_1) = \mathcal{C}_2$.

Heuristic analysis of the algorithm. The heuristic analysis of this algorithm is very similar to that of our permutation equivalence algorithm. This time the size of a M_2-orbit of a 2-space V with $|\mathrm{Supp}(V)| \leq w$ is bounded by $\frac{n!}{(n-w)!}(q-1)^{w-1}$ and a random 2-space has probability of $\frac{(q^k-1)(q^k-q)}{(q^n-1)(q^n-q)} \approx q^{2(k-n)}$ of being a subspace of $\mathcal{C}_1$. So, if we pick w such that $\frac{n!}{(n-w)!}q^{w-1+2k-2n} <$

$\frac{1}{4\log n}$ we expect the probability of finding a "bad" pair (V, W) (i.e. $\mathsf{lex}(V) = \mathsf{lex}(W)$ but $\mu(V) \neq W$) is bounded by $1/2$. The size of $X_1(w)$ and $X_2(w)$ is expected to be at most $\binom{n}{w}\frac{(q^w-1)(q^w-q)}{(q^2-1)(q^2-q)}q^{-2(n-k)} \approx \binom{n}{w}q^{2(w-2-n+k)}$, because for each of the $\binom{n}{w}$ supports S of size w, there are $\frac{(q^w-1)(q^w-q)}{(q^2-1)(q^2-q)}$ 2-spaces whose support is included in S, and we expect one out $q^2(n-k)$ of them to lie in $\mathcal{C}_1$. Therefore, if we set the list size to $\sqrt{\binom{n}{w} \cdot q^{2(-n+k+w-2)} \cdot 2\log n}$ then we expect the third step of the algorithm to terminate after roughly $|L|$ iterations. (We are counting the subspaces V with $|\mathrm{Supp}(V)| < w$ multiple times, so $X_1(w)$ is slightly smaller than our estimate. This is not a problem, because it means that the third step will terminate slightly sooner than our analysis suggests.)

The complexity of the algorithm consists of the ISD effort to sample $|L|$ elements from $X_1(w)$ and $X_2(w)$ respectively, and the costs of computing lex. We have to compute lex an expected number of $2|L|$ times; once for each of the 2-spaces in the list L and once for each 2-space found in step 3. Since the number of row operations per lex is $O(q)$, the total cost of computing lex is $O(q|L||)$. To sample the 2-spaces we use asn adaptation of the Lee-Brickell algorithm: We repeatedly put a generator matrix of $\mathcal{C}_1$ in echelon form with respect to a random choice of pivot columns, and then we look at the span of any 2 out of k rows of the new matrix. Given the echelon form of the matrix, the support of these 2-spaces has size at most $n-k+2$, and if we are lucky the size of the support will be smaller than or equal to w. The complexity of this algorithm is very similar to that of the standard Lee-Brickell algorithm for finding codewords (see section 2.2).

For a 2-space $V \in X_1(w)$, the Lee-Brickell algorithm will find V if the random choice of pivots intersects $\mathrm{Supp}(V)$ in 2 positions, which happens with probability $P_\infty(n,k,w) = \binom{n-k}{w-2}\binom{k}{2}/\binom{n}{w}$. The cost per iteration is $O(k^2 + \binom{k}{2}) = O(k^2)$ row operations for the Gaussian elimination and for enumerating the 2-spaces, so the expected number of row operations until we find $|L|$ elements in $X_1(w)$ and $X_2(w)$ is

$$O\left(\frac{k^2\binom{n}{w}|L|}{\binom{n-k}{w-2}\binom{k}{2}|X_1(w)|}\right) \approx O\left(\frac{\sqrt{\binom{n}{w}\cdot\log n}}{\binom{n-k}{w-2}}q^{-w+2+n-k}\right).$$

4.1 The algorithm in practice.

We have implemented the algorithm and applied it to the LESS-I parameter set. The public key of a LESS-I signature consist of two linearly equivalent codes $\mathcal{C}_1$ and $\mathcal{C}_2$ chosen uniformly at random of length $n = 54$ and dimension $k = 27$ over $\mathbb{F}_{53}$. The largest value of w satisfying $\frac{n!}{(n-w)!}q^{w-1+2k-2n} < \frac{1}{4\log n}$ is $w = 28$, so we use the Lee-Brickell algorithm to generate $\sqrt{\binom{n}{w} \cdot q^{2(-n+k+w-2)} \cdot 2\log n}$ subspaces of $\mathcal{C}_1$ and $\mathcal{C}_2$ with support of size at most 28, which comes down to approximately 2800000 subspaces of each code. From our implementation we see that this takes on average about $2^{20.6}$ Gaussian eliminations, which matches the heuristic analysis of the previous section very well. The attack takes in total about $2^{30.9}$ row operations, which amounts to about 25 seconds on a laptop with an Intel i5-8400H CPU at 2.50GHz. Approximately 12 seconds are spent computing the spaces V, the remaining time is spent computing $\mathsf{lex}(V)$.

Remark 2. Similar to the Permutation equivalence case, it is possible to use a memoryless collision search algorithm to remove the large memory cost of the attack at essentially no runtime cost.

Comparison with Leon's algorithm and new parameters for LESS. We expect Leon's algorithm (using the Lee-Brickell algorithm to instantiate the ISD oracle) to require $2^{38.3}$ row operations, which is significantly more than the $2^{30.9}$ operations that our algorithm requires. Figure 3 shows the complexity of our algorithm and Leon's algorithm for increasing code length. If the size of the finite field increases linearly with the code length, then the gap between our algorithm and Leon's algorithm increases exponentially. In contrast, if the field size is fixed, then Leon's algorithm will eventually outperform our algorithm. Figure 4 shows that our algorithm exploits the large field size so well, that in some regimes increasing the field size hurts security. Therefore, when picking parameters for LESS, it is best not to pick a field size that is too big. To achieve 128 bits of security against our algorithm and Leon's algorithm one could use linearly equivalent codes of length 250 and dimension 125 over $\mathbb{F}_{53}$. This results in a signature size of 28.4 KB, more than 3 times the original LESS-I signature size of 8.4 KB. The public key size would be 11.4 KB, more than 22 times the original public key size of 0.5 KB. We found that for the LESS-II parameter set, the finite field $\mathbb{F}_7$ is too small for our algoritm to improve over Leon's algoritm, which we estimate would take about 2^{44} row operations.

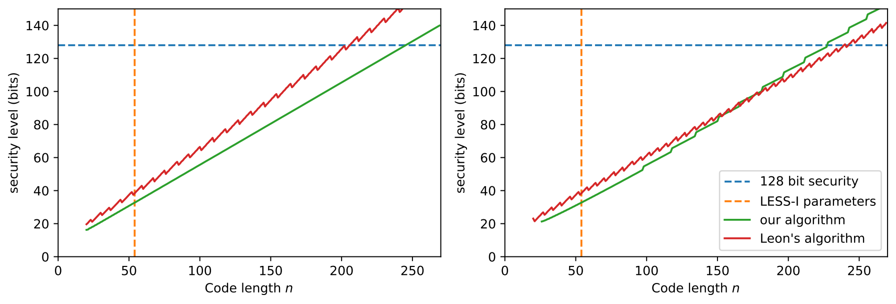


Fig. 3. Complexity of Leon's algorithm and our algorithm for finding linear equivalences in function of the code Length. In the left graph the field size scales linearly with the code length, in the right graph the field size $q = 53$ is fixed. In both cases the rate of the code is fixed at $k/n = 1/2$.

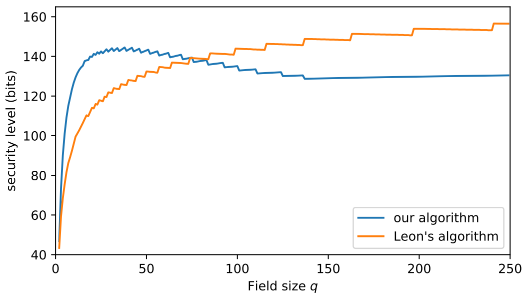


Fig. 4. Estimated complexity of Leon's algorithm and our algorithm for finding linear equivalences in function of the finite field size for random weakly self-dual codes of length $n = 250$ and dimension $k = 125$.

5 Conclusion

We have introduced a new algorithm for finding permutation equivalences and linear equivalences between codes that improves upon Leon's algorithm for sufficiently large field sizes. Leon's algorithm requires computing the set of all the codewords of minimal length, in contrast, to find permutation equivalences our algorithm only requires to compute a small (square root) fraction of the codewords that have a certain (non-minimal) weight. To find linear equivalences we compute a small fraction of the 2-dimensional subspaces of the code with small (but not minimal) support. We implement the algorithm and use it to break the recently proposed LESS system. We show that the LESS-I and LESS-III parameter sets can be broken in only 25 seconds and 2 seconds respectively. We propose larger parameters that resist our attack and Leon's original attack that come at the cost of at least a factor 3 increase in signature size and a factor 22 increase in key size. We compare the new parameters of LESS to some other code-based signature schemes in table 2. Despite the significant increase in signature size and key size, LESS still has smaller signatures than other zero-knowledge based signatures in the Hamming metric, such as Stern's protocol [15], Veron's protocol [17] and the CVE scheme [4]. For example, we estimate that with some straightforward optimizations, the Fiat-Shamir transformed version of CVE identification protocol has a signature size of 38 KB at 128 bits of security. However, the smaller signature size of LESS comes at the cost of larger public keys. Compared to cRVDC [2], a recent zero-knowledge-based proposal using the rank metric, the signature size of LESS is very similar, but the LESS public keys are much larger. Compared to the Durandal scheme [1], LESS has a similar public key size, but larger signatures. Finally, compared to WAVE [5] LESS has much smaller public keys, but also much larger signatures.

	CVE [15]	cRVDC [2]	Durandal [1]	Wave [5]	LESS-I	LESS-III
Metric Type	Hamming FS	Rank FS	Rank FS w/ abort	Hamming Trapdoor	Hamming FS	
Public Key	104 B	152 B	15 KB	3.2 MB	11 KB	17 KB
Signature	38 KB	22 KB	4.0 KB	1.6 KB	28 KB	19 KB

Table 2. Comparison of the new LESS parameters with some other code-based signature schemes.

References

1. Nicolas Aragon, Olivier Blazy, Philippe Gaborit, Adrien Hauteville, and Gilles Zémor. Durandal: A rank metric based signature scheme. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 728–758. Springer, Heidelberg, May 2019.
2. Emanuele Bellini, Florian Caullery, Philippe Gaborit, Marc Manzano, and Victor Mateu. Improved veron identification and signature schemes in the rank metric. In *2019 IEEE International Symposium on Information Theory (ISIT)*, pages 1872–1876. IEEE, 2019.
3. Jean-Francois Biasse, Giacomo Micheli, Edoardo Persichetti, and Paolo Santini. LESS is more: Code-based signatures without syndromes. Cryptology ePrint Archive, Report 2020/594, 2020. `https://eprint.iacr.org/2020/594`.
4. Pierre-Louis Cayrel, Pascal Véron, and Sidi Mohamed El Yousfi Alaoui. A zero-knowledge identification scheme based on the q-ary syndrome decoding problem. In Alex Biryukov, Guang Gong, and Douglas R. Stinson, editors, *SAC 2010*, volume 6544 of *LNCS*, pages 171–186. Springer, Heidelberg, August 2011.
5. Thomas Debris-Alazard, Nicolas Sendrier, and Jean-Pierre Tillich. Wave: A new family of trapdoor one-way preimage sampleable functions based on codes. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part I*, volume 11921 of *LNCS*, pages 21–51. Springer, Heidelberg, December 2019.
6. Thomas Feulner. The automorphism groups of linear codes and canonical representatives of their semilinear isometry classes. *Adv. in Math. of Comm.*, 3(4):363–383, 2009.
7. Marc Girault. A (non-practical) three-pass identification protocol using coding theory. In Jennifer Seberry and Josef Pieprzyk, editors, *AUSCRYPT'90*, volume 453 of *LNCS*, pages 265–272. Springer, Heidelberg, January 1990.
8. Pil Joong Lee and Ernest F. Brickell. An observation on the security of McEliece's public-key cryptosystem. In C. G. Günther, editor, *EUROCRYPT'88*, volume 330 of *LNCS*, pages 275–280. Springer, Heidelberg, May 1988.
9. Jeffrey Leon. Computing automorphism groups of error-correcting codes. *IEEE Transactions on Information Theory*, 28(3):496–511, 1982.
10. Robert J McEliece. A public-key cryptosystem based on algebraic coding theory. *Jet Propulsion Laboratory DSN Progress Report 42–44*, 1978.
11. Mohamed Ahmed Saeed. *Algebraic Approach for Code Equivalence*. PhD thesis, Normandie Université; University of Khartoum, 2017.
12. Nicolas Sendrier. Finding the permutation between equivalent linear codes: The support splitting algorithm. *IEEE Transactions on Information Theory*, 46(4):1193–1203, 2000.
13. Nicolas Sendrier and Dimitris E. Simos. The hardness of code equivalence over and its application to code-based cryptography. In Philippe Gaborit, editor, *Post-Quantum Cryptography - 5th International Workshop, PQCrypto 2013*, pages 203–216. Springer, Heidelberg, June 2013.
14. Jacques Stern. A method for finding codewords of small weight. In *International Colloquium on Coding Theory and Applications*, pages 106–113. Springer, 1988.
15. Jacques Stern. A new identification scheme based on syndrome decoding. In Douglas R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 13–21. Springer, Heidelberg, August 1994.

16. Paul C. van Oorschot and Michael J. Wiener. Parallel collision search with cryptanalytic applications. *Journal of Cryptology*, 12(1):1–28, January 1999.
17. Pascal Véron. Improved identification schemes based on error-correcting codes. *Applicable Algebra in Engineering, Communication and Computing*, 8(1):57–69, 1997.

Chapter 11

Cryptanalysis of UOV and Rainbow

Hell hath no fury like an ego scorned.

– Frederik Vercauteren

Publication Data

Beullens, W. Improved Cryptanalysis of UOV and Rainbow In: Advances in Cryptology – EUROCRYPT 2021. Lecture Notes in Computer Science. Springer, 2021. (Proceedings not yet published at time of writing)

Improved cryptanalysis of UOV and Rainbow

Ward Beullens

imec-COSIC, KU Leuven, Belgium

Abstract. The contributions of this paper are twofold. First, we simplify the description of the Unbalanced Oil and Vinegar scheme (UOV) and its Rainbow variant, which makes it easier to understand the scheme and the existing attacks. We hope that this will make UOV and Rainbow more approachable for cryptanalysts. Second, we give two new attacks against the UOV and Rainbow signature schemes; the intersection attack that applies to both UOV and Rainbow and the rectangular MinRank attack that applies only to Rainbow. Our attacks are more powerful than existing attacks. In particular, we estimate that compared to previously known attacks, our new attacks reduce the cost of a key recovery by a factor of 2^{17}, 2^{53}, and 2^{73} for the parameter sets submitted to the second round of the NIST PQC standardization project targeting the security levels I, III, and V respectively. For the third round parameters, the cost is reduced by a factor of 2^{20}, 2^{40}, and 2^{55} respectively. This means all these parameter sets fall short of the security requirements set out by NIST.

1 Introduction

The Oil and Vinegar scheme and its Rainbow variant are two of the oldest and most studied signature schemes in multivariate cryptography. The Oil and Vinegar scheme was proposed by Patarin in 1997 [17]. Soon thereafter, Kipnis and Shamir discovered that the original choice of parameters was weak and could be broken in polynomial time [15]. However, it is possible to pick parameters differently, such that the scheme resists the Kipnis-Shamir attack. This variant is called the Unbalanced Oil and Vinegar scheme (UOV), and has withstood all cryptanalysis since 1999 [14].

The rainbow signature scheme can be seen as multiple layers of UOV stacked on top of each other. This was proposed by Ding and Schmidt in 2005 [9]. The design philosophy is that by iterating the UOV construction, the Kipnis-Shamir

* This work was supported by CyberSecurity Research Flanders with reference number VR20192203, and by the Research Council KU Leuven grant C14/18/067 on Cryptanalysis of post-quantum cryptography. Ward Beullens is funded by FWO SB fellowship 1S95620N.

attack becomes less powerful, which enables the use of more efficient parameters. However, the additional complexity opened up more attack strategies, such as the MinRank attack, the Billet-Gilbert attack [4], and the Rainbow Band Separation attack [10]. Even though our understanding of the complexity of these attacks has been improving over the last decade, there have been no new attacks since 2008.

Multivariate cryptography is believed to resist attacks from adversaries with access to large scale quantum computers, which is why there has been renewed interest in this field of research during recent years. Seven out of the nineteen signature schemes that were submitted to the NIST post-quantum cryptography standardization project were multivariate signature schemes. From those seven schemes, four were allowed to proceed to the second round [3, 5, 18, 8], and only the Rainbow submission was selected as a finalist. The UOV scheme was not submitted to the NIST PQC project.

Contributions. As a first contribution, we simplify the description of the UOV and Rainbow schemes. Traditionally, the public key is a multivariate quadratic map $\mathcal{P}$, and the secret key is a factorization $\mathcal{P} = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T}$ where $\mathcal{S}$ and $\mathcal{T}$ are invertible linear maps, and $\mathcal{F}$ is a so-called central map. Our description avoids the use of a central map and only talks about properties of $\mathcal{P}$ instead. This new perspective makes it easier to understand the scheme and the existing attacks.

Secondly, we introduce two new key-recovery attacks: the intersection attack and the rectangular MinRank attack. The intersection attack relies on the idea behind the Kipnis-Shamir attack and applies to both the UOV scheme and the Rainbow scheme. The rectangular MinRank attack reduces key recovery to an instance of the MinRank problem. In this problem the task is, given a number of matrices, to find a linear combination of these matrices with exceptionally low rank. When Ding and Schmidt designed the Rainbow scheme in 2005 they were already aware that Rainbow was susceptible to MinRank attacks. However, our new attack shows that there was another instance of the MinRank problem lurking in the Rainbow public keys that went undiscovered until now. We call our attack the rectangular MinRank attack because unlike previous attacks, the matrices in the new MinRank instance are rectangular instead of square.

Roadmap. After giving some necessary background in Sect. 2, we introduce our simplified description of the Oil and Vinegar scheme and the existing attacks in Sect. 3. In Sect. 4 we introduce our intersection attack on UOV. In Sect. 5 we give a simplified description of the Rainbow scheme, and we review the existing attacks. The following sections 6 and 7 introduce the intersection attack for Rainbow and the rectangular MinRank attack respectively. We conclude in Sect. 8 with an overview of our attack complexities and new parameter sets for UOV and Rainbow.

2 Preliminaries

2.1 Notation.

For a vector space $V \subset K^n$ over a field K, we define its orthogonal complement $V^\perp$ as the space of vectors that are orthogonal to all the vectors in V, i.e. $V^\perp = \{\mathbf{w} | \langle \mathbf{w}, \mathbf{v} \rangle = 0\,, \forall \mathbf{v} \in V\}$. For a linear subspace $W \subset V$, we denote by V/W the quotient space of V by W. This is the vector space whose elements are the cosets of W in V:

$$V/W = \{\overline{\mathbf{x}} := \mathbf{x} + W \,|\, \mathbf{x} \in V\} \ .$$

Let $\mathbf{x} = x_1, \cdots, x_{n_x}$ and $\mathbf{y} = y_1, \cdots, y_{n_y}$ be two groups of variables in $\mathbb{F}_q$. We denote by $\mathcal{M}(a,b)$ the number of monomial functions of degree a in the $\mathbf{x}$ variables and degree b in the $\mathbf{y}$ variables. We denote by $\overline{\mathcal{M}}(a,b)$ the number of monomial functions of degree at most a in $\mathbf{x}$ and at most b in $\mathbf{y}$. If a and b are lower than q we have

$$\mathcal{M}(a,b) = \binom{a+n_x-1}{a}\binom{b+n_y-1}{b} \text{ and } \overline{\mathcal{M}}(a,b) = \binom{a+n_x}{a}\binom{b+n_y}{b}$$

2.2 Multivariate quadratic maps

The central object in Multivariate Quadratic cryptography is the multivariate quadratic map. A multivariate quadratic map $\mathcal{P}$ with m components and n variables is a sequence $p_1(\mathbf{x}), \cdots, p_m(\mathbf{x})$ of m multivariate quadratic polynomials in n variables $\mathbf{x} = (x_1, \cdots, x_n)$, with coefficients in a finite field $\mathbb{F}_q$.

To evaluate the map $\mathcal{P}$ at a value $\mathbf{a} \in \mathbb{F}_q^n$, we simply evaluate each of its component polynomials in $\mathbf{a}$ to get a vector $\mathbf{b} = (b_1 = p_1(\mathbf{a}), \cdots, b_m = p_m(\mathbf{a}))$ of m output elements. We denote this by $\mathcal{P}(\mathbf{a}) = \mathbf{b}$.

MQ problem The main source of computational hardness for multivariate cryptosystems is the Multivariate Quadratic (MQ) problem. Given a multivariate quadratic map $\mathcal{P} : \mathbb{F}_q^n \to \mathbb{F}_q^m$, and given a target $\mathbf{t} \in \mathbb{F}_q^m$, the MQ problem asks to find a solution $\mathbf{s}$ such that $\mathcal{P}(\mathbf{s}) = \mathbf{t}$. This problem is NP-hard, and it is believed to be exponentially hard on average, even for quantum adversaries. Currently, the best algorithms to solve instances of this problem (for cryptographically relevant parameters) are algorithms such as F_4/F_5 or XL that use a Gröbner-basis-like approach [11, 6].

Polar forms. For a multivariate quadratic polynomial $p(\mathbf{x})$, we can define its *polar form*

$$p'(\mathbf{x},\mathbf{y}) := p(\mathbf{x}+\mathbf{y}) - p(\mathbf{x}) - p(\mathbf{y}) + p(0)\,.$$

Similarly, for a multivariate quadratic map $\mathcal{P}(\mathbf{x}) = p_1(\mathbf{x}), \cdots, p_m(\mathbf{x})$, we define its polar form as $\mathcal{P}'(\mathbf{x},\mathbf{y}) = p_1'(\mathbf{x},\mathbf{y}), \cdots, p_m'(\mathbf{x},\mathbf{y})$. This polar form will allow us to simplify the descripion of the UOV and Rainbow schemes, and will play a major role in the attacks on UOV and Rainbow. The multivariate quadratic maps of interest in this paper are homogenous, so we will often omit the $\mathcal{P}(0)$ term.

Theorem 1. *Given a multivariate quadratic map $\mathcal{P}(\mathbf{x}) : \mathbb{F}_q^n \to \mathbb{F}_q^m$, its polar form $\mathcal{P}'(\mathbf{x},\mathbf{y}) : \mathbb{F}_q^n \times \mathbb{F}_q^n \to \mathbb{F}_q^m$ is a symmetric and bilinear map.*

Proof. We can write $p(\mathbf{x}) = \mathbf{x}^\top Q\mathbf{x} + \mathbf{v}\cdot\mathbf{x} + c$, where Q is an upper triangular matrix that contains the coefficients of the quadratic terms of p, where $\mathbf{v}$ contains the coefficients of the linear terms of $p(\mathbf{x})$, and where c is the constant term of $p(\mathbf{x})$. Then we have

$$\begin{aligned} p'(\mathbf{x},\mathbf{y}) &:= p(\mathbf{x}+\mathbf{y}) - p(\mathbf{x}) - p(\mathbf{y}) + p(0) \\ &= (\mathbf{x}+\mathbf{y})^\top Q(\mathbf{x}+\mathbf{y}) - \mathbf{y}^\top Q\mathbf{y} - \mathbf{x}^\top Q\mathbf{x} + \mathbf{v}\cdot(\mathbf{x}+\mathbf{y}) - \mathbf{v}\cdot\mathbf{x} - \mathbf{v}\cdot\mathbf{y} \\ &= \mathbf{x}^\top Q\mathbf{y} + \mathbf{y}^\top Q\mathbf{x} \\ &= \mathbf{x}^\top (Q + Q^\top)\mathbf{y}\,. \end{aligned}$$

□

2.3 Solving MinRank with Support Minors Modeling

The MinRank problem asks, given k matrices $L_1, \cdots, L_k$ with n rows and m columns and a target rank r, to find coefficients $y_i \in \mathbb{F}_q$ for i from 1 to k, not all zero, such that the linear combination $\sum_{i=1}^k y_i L_i$ has rank at most r.

Recently, Bardet *et al.* introduced the Support Minors Modeling algorithm for solving this problem [1]. Let $\mathbf{y} \in \mathbb{F}_q^k$ be a solution, and let C be a matrix whose rows form a basis for the rowspan of $L_\mathbf{y} = \sum_{i=1}^k y_i L_i$. For each subset $S \subset \{1, \cdots, m\}$ of size $|S| = r$, let c_S be the determinant of the r-by-r submatrix of C whose columns are the columns of C with index in S.

The Support Minors Modeling approach considers for each $j \in \{1, \cdots, n\}$ the matrix

$$C_j = \begin{pmatrix} r_j \\ C \end{pmatrix}\,,$$

where r_j is the j-th row of $L_\mathbf{y}$. Then the rank of C_j is at most r, which implies that all its $(r+1)$-by-$(r+1)$ minors vanish. Using cofactor expansion on the first row, each minor gives a bilinear equation in the y_i variables and the c_S variables. The Support Minors Modeling algorithm then uses the XL algorithm to find a solution to this system of $n\binom{m}{r+1}$ bilinear equations.

Analysis. The attack constructs the Macaulay matrix M_b at bi-degree $(b, 1)$, a large sparse matrix, whose columns correspond to the monomials of degree b in the y_i variables, and of degree 1 in the c_S variables. So at degree $(b, 1)$, the matrix has $\mathcal{M}(b, 1)$ columns. The rows of the matrix contain the degree $(b, 1)$ polynomials of the form $\mu(\mathbf{y}) \cdot f(\mathbf{y}, \mathbf{c})$, where $\mu(\mathbf{y})$ is a monomial of degree $b-1$, and $f(\mathbf{y}, \mathbf{c})$ is one of the bilinear equations of the Support Minors Modeling system. The goal of the attack is then to use the Wiedemann algorithm to find a non-trivial solution to the linear system $M_b\mathbf{x} = 0$, so that $\mathbf{x}$ reveals a solution to the MinRank problem. This approach works if the rank of M_b is $\mathcal{M}(b, 1) - 1$, so that there is only a one-dimensional solution space that corresponds to the unique (up to a scalar) solution of the MinRank problem.

Bardet *et al.* calculate that whenever $b < r + 2$, the rank of the Macaulay matrix is

$$R_{k,n,m,r}(b) = \sum_{i=1}^{b} (-1)^{i+1} \binom{m}{r+i} \binom{n+i-1}{i} \binom{k+b-i-1}{b-i}, \tag{1}$$

unless $R_{k,n,m,r}(b') > \mathcal{M}(b', 1) - 1$ for some $b' \leq b$, in which case the rank is equal to $\mathcal{M}(b, 1) - 1$. This allows to calculate for which b the attack will succeed.

If b_{min} is the smallest integer for which the attack will succeed, then solving the XL system with the Wiedemann algorithm requires

$$3\mathcal{M}(b_{min}, 1)^2(r + 1)k$$

field multiplications. Bardet *et al.* found that it is often advantageous to ignore a number of columns of the L_i matrices and only consider the first m' columns of the matrices, for some optimal value of m' in the range $[r + 1, m]$. For more details on the Support Minors Modeling algorithm, we refer to [1].

3 The UOV signature scheme

The Oil and Vinegar signature scheme, introduced in 1997 by Patarin [17], is based on an elegant MQ-based trapdoor function. The trapdoor function is a multivariate quadratic map $\mathcal{P} : \mathbb{F}_q^n \to \mathbb{F}_q^m$ for which it is assumed that finding preimages (i.e. solving the MQ problem) is hard. However, if one knows some extra information (called the trapdoor), then it is easy to find preimages for any arbitrary output. Originally, Patarin proposed to use the system with $n = 2m$.

This parameter choice was cryptanalysed by Kipnis and Shamir [15], which is why current proposals use $n > 2m$. This is known as the Unbalanced Oil and Vinegar (UOV) signature scheme. The conservative recommendation is to use $n = 3m$ or even $n = 4m$, but more aggressive and (more efficient) parameter sets have been proposed that use $n \approx 2.35m$ [7].

The UOV signature scheme is created from the UOV trapdoor function with the Full Domain Hash approach: The public key is the trapdoor function $\mathcal{P} : \mathbb{F}_q^n \to \mathbb{F}_q^m$, the secret key contains the trapdoor information, and a signature on a message M is simply an input $\mathbf{s}$ such that $\mathcal{P}(\mathbf{s}) = \mathcal{H}(M||\mathsf{salt})$, where $\mathcal{H}$ is a cryptographic hash function that outputs elements in the range of $\mathcal{P}$ and where salt is a fixed-length bit string chosen uniformly at random for every signature. Therefore, to understand the UOV signature scheme, we only need to understand how the UOV trapdoor function works.

3.1 UOV trapdoor function

The UOV trapdoor function is a multivariate quadratic map $\mathcal{P} : \mathbb{F}_q^n \to \mathbb{F}_q^m$ that vanishes on a secret linear subspace $O \subset \mathbb{F}_q^n$ of dimension $\dim(O) = m$, i.e.

$$\mathcal{P}(\mathbf{o}) = 0 \quad \text{for all } \mathbf{o} \in O\,.$$

The trapdoor information is nothing more than a description of O. To generate the trapdoor function one first picks the subspace O uniformly at random and then one picks $\mathcal{P}$ uniformly at random from the set of multivariate quadratic maps with m components in n variables that vanish on O. Note that on top of the q^m “artificial” zeros in the subspace O, we expect roughly q^{n-m} “natural” zeros that do not lie in O.

Given a target $\mathbf{t} \in \mathbb{F}_q^m$, how do we use this trapdoor to find $\mathbf{x} \in \mathbb{F}_q^n$ such that $\mathcal{P}(\mathbf{x}) = \mathbf{t}$? To do this, one picks a vector $\mathbf{v} \in \mathbb{F}_q^n$ and solves the system $\mathcal{P}(\mathbf{v} + \mathbf{o}) = \mathbf{t}$ for a vector $\mathbf{o} \in O$. This can simply be done by solving a linear system for $\mathbf{o}$, because

$$\mathcal{P}(\mathbf{v}+\mathbf{o}) = \underbrace{\mathcal{P}(\mathbf{v})}_{\text{fixed by choice of } \mathbf{v}} + \underbrace{\mathcal{P}(\mathbf{o})}_{=0} + \underbrace{\mathcal{P}'(\mathbf{v},\mathbf{o})}_{\text{linear function of } \mathbf{o}} = \mathbf{t}\,.$$

With probability roughly $1 - 1/q$ over the choice of $\mathbf{v}$ the linear map $\mathcal{P}'(\mathbf{v}, \cdot)$ will be non-singular, in which case the linear system $\mathcal{P}(\mathbf{v}+\mathbf{o}) = \mathbf{t}$ has a unique solution. If this is not the case, one can simply pick a new value for $\mathbf{v}$ and try again.

3.2 Traditional description of UOV

Traditionally, the UOV signature is described as follows: The secret key is a pair $(\mathcal{F}, \mathcal{T})$, where $\mathcal{T} \in GL(n, q)$ is a random invertible linear map, and $\mathcal{F} : \mathbb{F}_q^n \to \mathbb{F}_q^m$ is the so-called central map, whose components $f_1, \cdots, f_m$ are chosen uniformly at random of the form

$$f_i(\mathbf{x}) = \sum_{i=1}^{n} \sum_{j=i}^{n-m} \alpha_{i,j} x_i x_j \,.$$

Note that the second sum only runs from i to $n - m$. So all the terms have at least one variable in $x_1, \cdots, x_{n-m}$.

The public key that corresponds to $(\mathcal{F}, \mathcal{T})$ is the multivariate quadratic map $\mathcal{P} = \mathcal{F} \circ \mathcal{T}$. To sign a message M, the strategy is to first solve for $\mathbf{s}' \in \mathbb{F}_q^n$ such that $\mathcal{F}(\mathbf{s}') = \mathcal{H}(M||\mathsf{salt})$, and then the final signature is $\mathbf{s} = \mathcal{T}^{-1}(\mathbf{s}')$, such that $\mathcal{P}(\mathbf{s}) = \mathcal{F}(\mathbf{s}') = \mathcal{H}(M||\mathsf{salt})$.

The description in Sect. 3.1 is just a slightly different way of thinking about the same scheme. In particular, the distribution of public keys for this signature scheme is the same: The central map $\mathcal{F}$ is chosen uniformly from the set of maps that vanish on the m-dimensional space of vectors O' that consists of all the vectors whose first $n-m$ entries are zero, i.e. $O' = \{\mathbf{v} \mid v_i = 0 \text{ for all } i \leq n-m\}$. After composing with $\mathcal{T}$, we get a public key $\mathcal{P} = \mathcal{F} \circ \mathcal{T}$ that vanishes on some secret linear subspace $O = \mathcal{T}^{-1}(O')$.

3.3 Attacks on UOV

A straightforward approach to attack the UOV signature scheme is to completely ignore the existence of the oil subspace and directly try to solve the system $\mathcal{P}(\mathbf{x}) = \mathcal{H}(M||\mathsf{salt})$ to produce a signature for the message M. This can be done with a Gröbner basis-like approach such as XL or F_4/F_5 [11, 6]. This is called a direct attack.

More interestingly, the attacker can first try to find the oil space O. After O is found, the attacker can sign any message as if he was a legitimate signer. Two attacks in the literature take this approach.

Reconciliation attack. The reconciliation attack was developed by Ding *et al.* as a stepping stone towards the Rainbow Band Separation (RBS) attack on Rainbow [10]. As an attack on UOV, the reconciliation attack is not very useful, since it never outperforms a direct attack on UOV for properly chosen parameters. Nevertheless, we describe the attack here, since it can also be seen as a precursor to our intersection attack of Sect. 4.

The attack tries to find a vector $\mathbf{o} \in O$ by solving the system $\mathcal{P}(\mathbf{o}) = 0$. We know that $\dim(O) = m$, so if we impose m affine constraints on the entries of $\mathbf{o}$, we still expect a unique solution $\mathbf{o} \in O$.

If $n - m \leq m$, then we expect $\mathcal{P}(\mathbf{o}) = 0$ to have a unique solution after fixing m entries of $\mathbf{o}$. This is a system of m equations in fewer than m variables, so solving this system is more efficient than a direct attack.

If $n - m > m$ then $\mathcal{P}(\mathbf{o}) = 0$ will have a lot of solutions, only one of which corresponds to an $\mathbf{o} \in O$. Enumerating all the solutions is too costly, and the attack will not outperform a direct attack. We can try to solve the following system to find multiple vectors $\mathbf{o}_1, \cdots, \mathbf{o}_k$ in O simultaneously:

$$\begin{cases} \mathcal{P}(\mathbf{o}_i) = 0 & \forall i \in \{1, \cdots, k\} \\ \mathcal{P}'(\mathbf{o}_i, \mathbf{o}_j) = 0 & \forall i < j \in \{1, \cdots, k\} \end{cases} .$$

However, this increases the number of variables that appear in the system, and therefore the attack will usually not outperform a direct attack.

Once a first vector in O is found, finding subsequent vectors is much easier. If $\mathbf{o}$ is the first vector that we found, then a second vector $\mathbf{o}' \in O$ will satisfy

$$\begin{cases} \mathcal{P}(\mathbf{o}') = 0 \\ \mathcal{P}'(\mathbf{o}, \mathbf{o}') = 0 \end{cases} ,$$

which means we get m linear equations on $\mathbf{o}'$ for free. Therefore, the complexity of the attack is dominated by the complexity of finding the first vector in O.

Kipnis-Shamir attack. Historically, the first attack on the OV signature scheme was given by Kipnis and Shamir [15]. The basic version of this attack works when $n = 2m$, which was the case for the parameter sets initially proposed by Patarin.

Attack if $\mathbf{n = 2m}$. The attack looks at the m components of $\mathcal{P}'(\mathbf{x}, \mathbf{y})$. Each component $p_i'(\mathbf{x}, \mathbf{y}) = p_i(\mathbf{x} + \mathbf{y}) - p_i(\mathbf{x}) - p_i(\mathbf{y})$, defines a matrix M_i such that $p_i'(\mathbf{x}, \mathbf{y}) = \mathbf{x}^\top M_i \mathbf{y}$. Kipnis and Shamir observed the following useful property of M_i.

Lemma 2. *For each $i \in \{1, \cdots, m\}$, we have that $M_i O \subset O^\perp$. That is, each M_i sends O into its own orthogonal complement $O^\perp$.*

Proof. For any $\mathbf{o}_1, \mathbf{o}_2 \in O$ we need to prove that $\langle \mathbf{o}_2, M_i \mathbf{o}_1 \rangle = 0$. This follows from the assumption that p_i vanishes on O:

$$\langle \mathbf{o}_2, M_i \mathbf{o}_1 \rangle = \mathbf{o}_2^\top M_i \mathbf{o}_1 = p_i'(\mathbf{o}_1, \mathbf{o}_2) = p_i(\mathbf{o}_1 + \mathbf{o}_2) - p_i(\mathbf{o}_1) - p_i(\mathbf{o}_2) = 0\,. \quad \square$$

If $n = 2m$, then $\dim(O^\perp) = n - m = m$, so if M_i is nonsingular (which happens with high probability[1]), then Lemma 2 turns into an equality $M_i O = O^\perp$. This means that for any pair of invertible M_i, M_j, we have that $M_j^{-1} M_i O = O$, i.e. that O is an invariant subspace of $M_j^{-1} M_i$. It turns out that finding a common invariant subspace of a large number of linear maps can be done in polynomial time, so this gives an efficient algorithm for finding O. For more details we refer to [15]

Remark 3. Note that, as a map from $\mathbb{F}_q^n$ to itself, M_i implicitly depends on a choice of basis for $\mathbb{F}_q^n$. A more natural approach would be to define M_i as a map from $\mathbb{F}_q^n$ to its dual $\mathbb{F}_q^{n\vee}$ given by $\mathbf{x} \mapsto p_i'(\mathbf{x}, \cdot)$. Lemma 2 would then say $M_i O \subset O^0$, where $O^0 \subset \mathbb{F}_q^{n\vee}$ is the annihilator of O. We chose not to take this approach to avoid the dual vector space and annihilators, which some readers might not be familiar with.

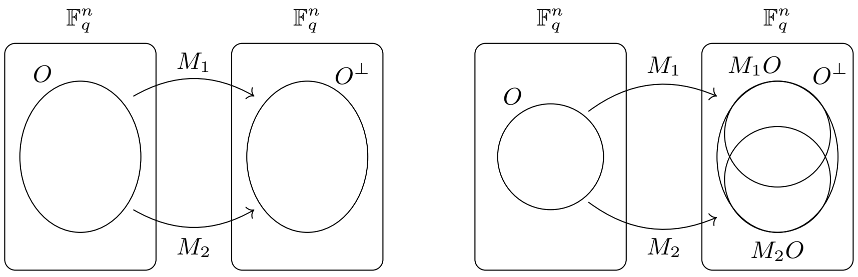


Fig. 1. Behavior of O under M_1 and M_2, in case $n = 2m$ (on the left) and $2m < n < 3m$ (on the right).

Attack if n > 2m. If $n > 2m$, then it is still the case that M_i sends O into $O^\perp$, but because $\dim(O^\perp) = n - m > m = \dim(O)$ the equality $M_i O = M_j O$

[1] In fields of characteristic 2 and in case n is odd, the M_i are never invertible, because M_i is skew-symmetric and with zeros on the diagonal and therefore has even rank. (Recall that $M_i = Q_i + Q_i^\perp$ as in the proof of Theorem 1.) To avoid this case we can always set one of the variables to zero. This has the effect of reducing n by one (which gets us back to the case where n is even), and it also reduces the dimension of O by one, which makes the attack slightly less powerful. Since this trick is always possible, we will assume that n is even in the remainder of the paper.

may no longer hold. Therefore, $M_i^{-1}M_j$ is no longer guaranteed to have O as an invariant subspace and the basic attack fails. However, even though in general $M_iO \neq M_jO$, they still have an unusually large intersection (see Figure 1): M_iO and M_jO are both subspaces of $O^{\perp}$, so their intersection has dimension at least $\dim(M_iO)+\dim(M_jO)-\dim(O^{\perp}) = 3m-n$. Kipnis *et al.* [14] realized that this means that vectors in O are more likely to be eigenvectors of $M_j^{-1}M_i$.

Heuristically, for $\mathbf{x} \in O$, the probability that it gets mapped by M_i to some point in the intersection $M_iO \cap M_jO$ is approximately

$$\frac{|M_iO \cap M_jO|}{|M_iO|} = q^{2m-n} .$$

If this happens, then the probability that M_j^{-1} maps $M_i\mathbf{x}$ back to a multiple of $\mathbf{x}$ is expected to be $(q-1)/|O| \approx q^{1-m}$. Therefore, we can estimate that the probability that a vector in O is an eigenvector of $M_j^{-1}M_i$ is approximately q^{1+m-n}, and the expected number of eigenvectors in O is therefore q^{1+2m-n}.

The same analysis holds when you replace M_i and M_j by arbitrary invertible linear combinations of the M_i. The attacker can repeatedly compute the eigenvectors of $F^{-1}G$, where F and G are random invertible linear combinations of the M_i. After q^{n-2m} attempts he can expect to find a vector in O (he can verify whether a given eigenvector $\mathbf{x}$ is in O by checking that $\mathcal{P}(\mathbf{x}) = 0$). The complexity of the attack is $\tilde{O}(q^{n-2m})$, so the attack runs in polynomial time if $n = 2m$, but quickly becomes infeasible for unbalanced instances of the OV construction[2]. For more details on the attack, we refer to [14].

4 Intersection attack on UOV

In this section, we introduce a new attack that uses the ideas behind the Kipnis-Shamir attack, in combination with a system-solving approach such as in the reconciliation attack. We first describe a basic version of the attack that works as long as $n < 3m$. Then we also give a more efficient version of the attack that works if $n < 2.5m$.

4.1 Attack if $n < 3m$

Like in the Kipnis-Shamir attack, we consider for each $i \in \{1, \cdots, m\}$ the matrix M_i such that $p'_i(\mathbf{x}, \mathbf{y}) = \mathbf{x}^\top M_i \mathbf{y}$, and we choose two indices $i, j \in \{1, \cdots, m\}$ such that M_i and M_j are invertible matrices. The goal of our attack is to find a vector $\mathbf{x}$ in the intersection $M_iO \cap M_jO$. Recall from Sect. 3.3

[2] The $\tilde{O}$-notation ignores polynomial factors.

that this intersection has dimension at least $3m - n$, so non-trivial solutions exist if $n < 3m$.

If $\mathbf{x}$ is in the intersection $M_iO \cap M_jO$, then both $M_i^{-1}\mathbf{x}$ and $M_j^{-1}\mathbf{x}$ are in O. Therefore, $\mathbf{x}$ is a solution to the following system of quadratic equations

$$\begin{cases} \mathcal{P}(M_i^{-1}\mathbf{x}) = 0 \\ \mathcal{P}(M_j^{-1}\mathbf{x}) = 0 \\ \mathcal{P}'(M_i^{-1}\mathbf{x}, M_j^{-1}\mathbf{x}) = 0 \end{cases} . \tag{2}$$

Since there is a $3m-n$ dimensional subspace of solutions, we can impose $3m-n$ affine constraints on $\mathbf{x}$, so that we expect a unique solution. The attack is then to simply use the XL algorithm to find a solution to this system of $3m$ quadratic equations in $n - (3m - n) = 2n - 3m$ variables.

Once $\mathbf{x}$ is found, we know 2 vectors $M_i^{-1}\mathbf{x}$ and $M_j^{-1}\mathbf{x}$ in O, and the remaining vectors in O can be found more easily with the approach described in Sect. 3.3.

4.2 Attack when $n < 2.5m$

If n is small enough compared to m we can make the attack more efficient by solving for an $\mathbf{x}$ in the intersection of more than 2 subspaces M_iO at the same time. Suppose $n < \frac{2k-1}{k-1}m$ for an integer $k \geq 1$, and let $L_1, \cdots, L_k$ be k randomly chosen invertible linear combinations of the M_i, then the intersection $L_1O \cap \cdots \cap L_kO$ will have dimension at least $km - (k-1)(n-m) > 0$, which means there is a nonzero $\mathbf{x}$ such that $L_i^{-1}\mathbf{x} \in O$ for all i from 1 to k. We can then solve the following system of equations:

$$\begin{cases} \mathcal{P}(L_i^{-1}\mathbf{x}) = 0\,, & \forall i \in \{1, \cdots, k\} \\ \mathcal{P}'(L_i^{-1}\mathbf{x}, L_j^{-1}\mathbf{x}) = 0\,, & \forall i < j \in \{1, \cdots, k\} \end{cases} \tag{3}$$

We expect to find a unique solution after imposing $km - (k-1)(n-m)$ linear conditions on $\mathbf{x}$ to random values, so the complexity of the attack is dominated by the complexity of solving a system of $\binom{k+1}{2}m$ quadratic equations in $nk - (2k-1)m$ variables.

Remark 4. Note that in the case $n = 2m$ the requirement $n < \frac{2k-1}{k-1}m$ is satisfied for every $k > 1$. If we pick $k \approx \sqrt{m}$, then we have more than $\binom{m+1}{2}$ equations in m variables, which means we can linearize the system and solve it with Gaussian elimination in polynomial time. This is not surprising, because Kipnis and Shamir have already shown that UOV can be broken in polynomial time if $n = 2m$.

4.3 Complexity analysis of the attack

We noticed that the equations of system (3) are not linearly independent: even though there are $\binom{k+1}{2}m$ equations they only span a subspace of dimension $\binom{k+1}{2}m - 2\binom{k}{2}$. This is because if we have $L_i = \sum_{l=1}^{m} \alpha_{il} M_i$, for all i from 1 to k, then for all $1 \leq i < j \leq k$ we have

$$\begin{aligned}\sum_{l=1}^{m} \alpha_{il} \mathcal{P}'_l(L_i^{-1}\mathbf{x}, L_j^{-1}\mathbf{x}) &= \sum_{l=1}^{m} \alpha_{il} (L_i^{-1}\mathbf{x})^{\perp} M_l L_j^{-1}\mathbf{x}) \\ &= (L_i^{-1}\mathbf{x})^{\perp} L_i L_j^{-1}\mathbf{x}) \\ &= \mathbf{x}^{\perp} L_j^{-1}\mathbf{x} = \sum_{l=1}^{m} \alpha_{jl} \mathcal{P}_l(L_j^{-1}\mathbf{x})\end{aligned}$$

Similarly, we have

$$\sum_{l=1}^{m} \alpha_{jl} \mathcal{P}'_l(L_i^{-1}\mathbf{x}, L_j^{-1}\mathbf{x}) = \mathbf{x}^{\perp} L_i^{-1}\mathbf{x} = \sum_{l=1}^{m} \alpha_{il} \mathcal{P}_l(L_i^{-1}\mathbf{x}) \,,$$

so for each choice of $0 \leq i < j \leq k$ there are two linear dependencies between the equations of system (3). This explains why they only span a subspace of dimension $\binom{k+1}{2}m - 2\binom{k}{2}$.

Our experiments show that, after removing the $2\binom{k}{2}$ redundant equations, the systems (2) and (3) behave like random systems of $M = \binom{k+1}{2}m - 2\binom{k}{2}$ quadratic equations in $N = nk - (2k-1)m$ variables. For some small UOV systems, we computed the ranks of the Macaulay matrices at various degrees, and we found that they exactly match the ranks of generic systems (see Table 1). That is, at degree d, the rank is equal to the coefficient of t^d in the power series expansion of

$$\frac{1-(1-t^2)^M}{(1-t)^{N+1}} \,,$$

assuming that this coefficient does not exceed the number of columns of the Macaulay matrix.

We can use the standard methodology for estimating the complexity of system solving with an XL Wiedemann approach as

$$3\binom{N+d_{reg}}{d_{reg}}^2 \binom{N+2}{2}$$

field multiplications, where the degree of regularity d_{reg} is the first d such that the coefficient of t^d in

$$\frac{(1-t^2)^M}{(1-t)^{N+1}}$$

is non-positive [2, 8].

Table 1. The rank and the number of columns of the Macaulay matrices for the system of equations of the intersection attack. The rank at degree d always matches the coefficients of t^d the corresponding generating function, except if the coefficient is larger or equal to the number of columns. In this case (marked by boldface in the table) the rank equals the number of columns minus 1, and the XL system can be solved at that degree d.

parameters				Macaulay matrix at degree d				Generating
n	m	k		$d=2$	$d=3$	$d=4$	$d=5$	function
8	4	2	Rank	10	**34**			$\frac{1-(1-t^2)^{10}}{(1-t)^5}$
			#Columns	15	35			
10	4	2	Rank	10	90	405	1245	$\frac{1-(1-t^2)^{10}}{(1-t)^9}$
			#Columns	45	165	495	1287	
12	5	2	Rank	13	130	673	**2001**	$\frac{1-(1-t^2)^{13}}{(1-t)^{10}}$
			#Columns	55	220	715	2002	
12	5	3	Rank	24	288	**1364**		$\frac{1-(1-t^2)^{24}}{(1-t)^{12}}$
			#Columns	78	364	1365		
14	6	2	Rank	16	176	936	**3002**	$\frac{1-(1-t^2)^{16}}{(1-t)^{11}}$
			#Columns	66	286	1001	3003	
14	6	3	Rank	30	390	**1819**		$\frac{1-(1-t^2)^{30}}{(1-t)^{13}}$
			#Columns	91	455	1820		

Concrete costs To demonstrate that the new attack is more efficient than existing attacks, we apply it to the UOV parameters proposed by Czypek *et al.* [7]. They proposed to use $q = 256, n = 103, m = 44$, targeting 128 bits of security. More precisely, they estimate that the direct attack requires 2^{130} field multiplications and that the Kipnis-Shamir attack requires 2^{136} multiplications.

Their parameter choice satisfies $n < 2.5m$, so we can use the more efficient version of the attack with $k = 3$ (i.e. where we solve for $\mathbf{x}$ in the intersection of 3 subspaces of the form M_iO). This results in a system of $M = \binom{3+1}{2}m - 2\binom{3}{2} = 258$ equations in $N = nk - (2k-1)m = 89$ variables. The complexity of finding a solution is 2^{95} multiplications ($d_{reg} = 9$), which is lower than the claimed security level of 2^{128} multiplications.

In general, it seems that the new attack only outperforms a direct forgery attack, if $n < 2.5$. The usual recommendation in the literature is to use $n = 3m$

or even $n = 4m$, so these parameters are not affected by the new attack. In contrast, the example above shows that more aggressive parameters (which are tempting because they are much more efficient and previously no attacks were known) are no longer secure.

5 The Rainbow signature scheme

The Rainbow signature scheme is a variant of the UOV signature scheme proposed in 2004 by Ding and Schmidt [9]. The Rainbow trapdoor function is a multivariate quadratic map $\mathcal{P} : \mathbb{F}_q^n \to \mathbb{F}_q^m$. The trapdoor consists of a sequence of nested subspaces $\mathbb{F}_q^n \supset O_1 \supset \cdots \supset O_l$ of the input space, and a sequence of nested subspaces $\mathbb{F}_q^m \supset W_1 \supset \cdots \supset W_l = \{0\}$ of the output space, with $\dim O_1 = m$, and $\dim O_i = \dim W_{i-1}$ for $i > 1$ and such that the following hold:

1. $\mathcal{P}(\mathbf{x}) \in W_i$ for all $\mathbf{x} \in O_i$, and
2. $\mathcal{P}'(\mathbf{x}, \mathbf{y}) \in W_{i-1}$ for all $\mathbf{x} \in \mathbb{F}_q^n$, all $\mathbf{y} \in O_i$.

Rainbow with one layer (i.e. $l = 1$) is nothing more than UOV. In the rest of the paper, we focus on Rainbow with two layers (i.e. $l = 2$), because this results in the most efficient schemes and because this covers all the parameter sets submitted to the NIST PQC standardization project. In this case, there are 3 secret subspaces: O_1, O_2 and W (see Figure 2). An instantiation of Rainbow is then described by 4 parameters:

– q: the size of the finite field
– n: the number of variables
– m: the number of equations in the public key, also the dimension of O_1.
– o_2: the dimension of O_2, also the dimension of W.

Given the trapdoor information (i.e. O_1, O_2 and W), a solution $\mathbf{s}$ to $\mathcal{P}(\mathbf{s}) = \mathbf{t}$ can be found with an efficient 2-step algorithm.

1. In the first step, pick $\mathbf{v} \in \mathbb{F}_q^n$ uniformly at random, and solve for $\overline{\mathbf{o}}_1 \in O_1/O_2$, such that $\mathcal{P}(\mathbf{v} + \overline{\mathbf{o}}_1) + W = \mathbf{t} + W$. This can be rewritten as

$$\underbrace{\mathcal{P}(\mathbf{v})}_{\text{fixed by choice of } \mathbf{v}} + \underbrace{\mathcal{P}(\overline{\mathbf{o}}_1)}_{\in W} + \underbrace{\mathcal{P}'(\mathbf{v}, \overline{\mathbf{o}}_1)}_{\text{linear in } \mathbf{o}_1} + W = \mathbf{t} + W.$$

   This is a system of linear equations in the quotient space $\mathbb{F}_q^m/W$, so we can efficiently sample a solution with Gaussian elimination. Note that the system has $m - \dim W$ constraints and $m - \dim W$ degrees of freedom, so we expect there to be a unique solution (mod O_2) with probability approximately $1 - 1/q$. If there is no unique solution we pick a new value of $\mathbf{v}$ and start over.

$$\begin{tikzcd}
\mathbb{F}_q^n \arrow[d, "\mathcal{P}"'] & O_1 \arrow[l, hook'] \arrow[dl, "\mathcal{P}'(\mathbf{x},\cdot)"'] \arrow[d, "\mathcal{P}"] & O_2 \arrow[l, hook'] \arrow[dl, "\mathcal{P}'(\mathbf{x},\cdot)"'] \arrow[d, "\mathcal{P}"] \\
\mathbb{F}_q^m & W \arrow[l, hook'] & \{0\} \arrow[l, hook']
\end{tikzcd}$$

Fig. 2. The structure of a Rainbow public key with 2 layers. The polar form $\mathcal{P}'(\mathbf{x},\cdot)$ maps O_2 to W for every $\mathbf{x} \in \mathbb{F}_q^n$.

2. In the second step, we solve for $\mathbf{o}_2 \in O_2$, such that $\mathcal{P}(\mathbf{v}+\mathbf{o}_1+\mathbf{o}_2) = \mathbf{t}$. Writing it as

$$\underbrace{\mathcal{P}(\mathbf{v}+\mathbf{o}_1)-\mathbf{t}}_{\text{fixed},\in W}+\underbrace{\mathcal{P}(\mathbf{o}_2)}_{=0}+\underbrace{\mathcal{P}'(\mathbf{v}+\mathbf{o}_1,\mathbf{o}_2)}_{\text{linear in }\mathbf{o}_2,\in W}=0\,,$$

we see that this is a system of $\dim W$ linear equations (because all the values are in W) in $\dim W$ variables, so we expect to find a unique solution with Gaussian elimination with probability $1-1/q$. If no unique solution exists we return to step 1 with a new guess of $\mathbf{v}$.

Remark 5. If we put $W = \mathbb{F}_q^m$ and $O_1 = O_2$, or if we put $O_2 = \{0\}$ and $W = \{0\}$ then we get back the original UOV construction.

5.1 Traditional description of Rainbow

Traditionally, a Rainbow public key is generated as $\mathcal{P} = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T}$, where $\mathcal{S} \in GL(m,q)$ and $\mathcal{T} \in GL(n,q)$ are uniformly random invertible linear maps, and where $\mathcal{F}(\mathbf{x}) = f_1(\mathbf{x}),\cdots,f_m(\mathbf{x})$ is the so-called central map, whose first o_1 components $f_1(\mathbf{x}),\dots,f_{o_1}(\mathbf{x})$ are of the form

$$f_i(\mathbf{x}) = \sum_{j=1}^{n-o_1}\sum_{k=1}^{n-m} \alpha_{ijk}x_jx_k\,,$$

and whose remaining components $f_{o_1+1}(\mathbf{x}),\cdots,f_m(\mathbf{x})$ are of the form

$$f_i(\mathbf{x}) = \sum_{j=1}^{n}\sum_{k=1}^{n-o_1} \alpha_{ijk}x_jx_k\,.$$

Let O_1' be the subspace of $\mathbb{F}_q^n$ consisting of all the vectors whose first $n-m$ entries are zeros, and let O_2' be the subspace consisting of the vectors whose first $n-o_2$ entries are zero. Then all the polynomials in the central map vanish on O_2', and the first o_1 polynomials also vanish on O_1'. In other words, $\mathcal{F}(O_2') = 0$ and $\mathcal{F}(O_1') \subset W'$, where W' is the subspace of $\mathbb{F}_q^m$ consisting of the vectors whose first o_1 entries are zero. Moreover, $\mathcal{F}'(\mathbf{x},\mathbf{y}) \in W'$ for any $\mathbf{x} \in \mathbb{F}_q^n$ and any $\mathbf{y} \in O_2$. Therefore, the central map $\mathcal{F}$ satisfies the diagram in Figure 2 with the publicly known subspaces O_1', O_2' and W' taking the roles of O_1, O_2 and W. This means that after composing $\mathcal{F}$ with secret random linear maps $\mathcal{S}$ and $\mathcal{T}$ we obtain a public key $\mathcal{P} = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T}$ that satisfies the diagram in Figure 2 for uniformy random secret subspaces $O_1 = \mathcal{T}^{-1}O_1'$, $O_2 = \mathcal{T}^{-1}O_2'$ and $W = \mathcal{S}^{-1}W'$.

5.2 Rainbow NIST PQC parameter sets

In this paper, we focus on the Rainbow parameter sets that were proposed to the second round and the finals of the NIST PQC standardization project [8]. These parameter sets and the corresponding key and signature sizes are displayed in Table 2.

Table 2. The Rainbow parameter sets that were submitted to the second round and the finals of the NIST PQC standardization project.

Parameter set		Parameters q	n	m	o_2	|pk| (kB)	|sk| (kB)	|sig| (Bytes)
Second Round	Ia	16	96	64	32	149	93	64
	IIIc	256	140	72	36	710	511	156
	Vc	256	188	96	48	1705	1227	204
Finals	Ia	16	100	64	32	157	101	66
	IIIc	256	148	80	48	861	611	164
	Vc	256	196	100	64	1885	1376	212

5.3 Attacks on Rainbow

A straightforward method to forge a signature is to simply try to find a solution $\mathbf{s}$ to the system $\mathcal{P}(\mathbf{s}) = \mathcal{H}(M||\mathsf{salt})$. This is called a direct attack. More interesting attacks try to exploit the hidden structure of the Rainbow trapdoor.

OV attack. The OV attack of Kipnis and Shamir to find the subspace O in the OV construction can be used against Rainbow to find O_2. The complexity of the attack is $\tilde{O}(q^{n-2o_2})$.

When O_2 is found, it is easy to find W, because

$$\{\mathcal{P}'(\mathbf{x},\mathbf{y}) \mid \mathbf{x} \in \mathbb{F}_q^n, \mathbf{y} \in O_2\} \subset W\,,$$

and with overwhelming probability this will be an equality. Once W is found, we have reduced the problem to a small UOV instance with parameters $n' = n - o_2$ and $m' = m - o_2$, so the Kipnis and Shamir attack can be used again to find O_1, with complexity $\tilde{O}(q^{n'-2m'}) = \tilde{O}(q^{n+o_2-2m})$, which is negligible compared to the complexity of the first step.

MinRank/HighRank attack. For all $i \in \{1, \cdots, m\}$, we define $M_i \in \mathbb{F}_q^{n\times n}$ like we did in the description of the OV attack. For $\mathbf{v} \in \mathbb{F}_q^m$ we define the linear combination $M_\mathbf{v} := \sum_{i=0}^m v_i M_i$. Then it follows that $\langle \mathbf{v}, \mathcal{P}'(\mathbf{x},\mathbf{y})\rangle = \mathbf{x}^\top M_\mathbf{v}\mathbf{y}$. The second property of the Rainbow public key says that if $\mathbf{v} \in W^\perp$, then $\langle \mathbf{v}, \mathcal{P}'(\mathbf{x},\mathbf{y})\rangle = \mathbf{x}^\top M_\mathbf{v}\mathbf{y} = 0$ for all values of $\mathbf{x}$ and all $\mathbf{y} \in O_2$. This implies that O_2 is in the kernel of $M_\mathbf{v}$, so $M_\mathbf{v}$ has an exceptionally small rank of at most $n - \dim O_2$.

The MinRank attack attempts to exploit this property to find a vector in $W^\perp$. The problem is, given the M_i for $i \in \{1, \cdots, m\}$, to find a linear combination of these maps that has rank $n - \dim O_2$. This can be done with 2 strategies:

Guessing strategy [13]. Repeatedly pick $\mathbf{v} \in \mathbb{F}_q^m$. With probability q^{-o_2}, we have $\mathbf{v} \in W^\perp$. To check if a guess is correct, we simply check if the rank of $M_\mathbf{v}$ is at most $n - \dim O_2$. The complexity of the attack is $\tilde{O}(q^{o_2})$. There is a more efficient version of this attack by Billet and Gilbert, that runs in time $\tilde{O}(q^{2n-3m+o_2+1})$ [4].

Algebraic strategy. One expresses $\operatorname{rank}(M_\mathbf{v}) \leq n - \dim O_2$ as a system of multivariate polynomial equations in the entries of $\mathbf{v}$ and uses an algorithm such as XL to find a solution. There exist several methods to translate the rank condition into a system of polynomial equations, such as the Kipnis-Shamir modeling, and Minors modeling [16, 12]. Recently, a more efficient approach by Bardet *et al.* called "Support Minors Modeling" drastically improved the efficiency of this attack (see Sec. 2.3 and [1]). The algebraic approach is asymptotically more efficient than the guessing strategy.

As soon as a single vector $\mathbf{v} \in W^{\perp}$ is found, the attacker knows O_2, because it is the kernel of $M_{\mathbf{v}}$. Then, once O_2 is known he can finish the key recovery attack as described in the previous section on the UOV attack.

Rainbow band separation attack This attack, proposed by Ding *et al.* [10], tries to simultaneously find a vector $\mathbf{o} \in O_2$, and a vector $\mathbf{v} \in W^{\perp}$. This gives rise to the following system of equations

$$\begin{cases} \mathcal{P}(\mathbf{o}) = 0 \\ \langle \mathbf{v}, \mathcal{P}'(\mathbf{o}, \mathbf{x}) \rangle = 0\,, \quad \forall \mathbf{x} \in \mathbb{F}_q^n \end{cases} . \tag{4}$$

To get a unique solution, we can impose o_2 linear relations on the entries of $\mathbf{o}$ and $m - o_2$ linear relations on the entries of $\mathbf{v}$. This results in a system with $n - o_2$ variables for $\mathbf{o}$ and o_2 variables for $\mathbf{v}$, which makes a total of n variables. It looks like we get q^n bilinear equations (one for each choice of $\mathbf{x} \in \mathbb{F}_q^n$), but these equations are obviously not independent. Extend $\mathbf{o}$ to a basis $\mathbf{x}_1 = \mathbf{o}, \mathbf{x}_2, \cdots, \mathbf{x}_n$ for $\mathbb{F}_q^n$ (since we fixed some entries of $\mathbf{o}$, we can pick the $\mathbf{x}_i$ with $i > 1$ without having to know the precice value of $\mathbf{o}$). We can rewrite system (4) as

$$\begin{cases} \mathcal{P}(\mathbf{o}) = 0 \\ \langle \mathbf{v}, \mathcal{P}'(\mathbf{o}, \mathbf{x}_i) \rangle = 0\,, \quad \forall i \in \{1, \cdots, n\} \end{cases} . \tag{5}$$

Note that the first bilinear equation is $\langle \mathbf{v}, \mathcal{P}'(\mathbf{o}, \mathbf{o}) \rangle = 0$, which is equivalent to $\langle \mathbf{v}, \mathcal{P}(2\mathbf{o}) - 2\mathcal{P}(\mathbf{o}) \rangle = \langle \mathbf{v}, 2\mathcal{P}(\mathbf{o}) \rangle = 0$, (recall that $\mathcal{P}$ is homogenous), so this equation is already implied by the $\mathcal{P}(\mathbf{o}) = 0$ equations. This leaves us with a system of m quadratic equations in $\mathbf{o}$, and $n - 1$ bi-linear equations in the entries of $\mathbf{o}$ and $\mathbf{v}$. The complexity of this attack is studied in detail in [19], where they introduce a variant of the XL algorithm that exploits the bi-homogenous structure of the system.

6 Intersection attack on Rainbow

In this section we introduce a new key-recovery attack against the Rainbow signature scheme that is similar to our intersection attack on UOV from Sect. 4. Let k be such that $n < \frac{2k-1}{k-1} o_2$, and pick invertible matrices $L_1, \cdots, L_k$ from the span of the M_i. Our goal is to find a vector $\mathbf{x}$ in the intersection

$$\mathbf{x} \in \bigcap_{i=1}^{k} L_i O_2 \,.$$

This intersection has dimension at least $ko_2 - (k-1)(n - o_2) > 0$, so non-zero vectors in the intersection exist. We could try to find $\mathbf{x}$ by solving the system (3). However, similar to the RBS attack, we can improve the efficiency of the attack by simultaneously looking for a vector $\mathbf{v} \in W^{\perp}$. Let $\mathbf{e}_1, \cdots, \mathbf{e}_n$ be a basis for $\mathbb{F}_q^n$, where all the entries of $\mathbf{e}_i$ are zero, except the i-th entry which equals 1. Then we get the following system of quadratic equations:

$$\begin{cases} \mathcal{P}(L_i^{-1}\mathbf{x}) = 0\,, & \forall i \in \{1, \cdots, k\} \\ \mathcal{P}'(L_i^{-1}\mathbf{x}, L_j^{-1}\mathbf{x}) = 0\,, & \forall i < j \in \{1, \cdots, k\} \\ \langle \mathbf{v}, \mathcal{P}'(L_i^{-1}\mathbf{x}, \mathbf{e}_j)\rangle = 0\,, & \forall i \in \{1, \cdots, k\} \text{ and } \forall j \in \{1, \cdots, n\} \end{cases} . \tag{6}$$

If we impose $ko_2 - (k-1)(n-o_2)$ affine constraints on the entries of $\mathbf{x}$, and $m - o_2$ affine constraints on the entries of $\mathbf{v}$ we expect to have a unique solution.

It looks like we get $\binom{k+1}{2}m$ quadratic equations in the $\mathbf{x}$ variables and kn equations that are linear in the $\mathbf{x}$ variables and the $\mathbf{v}$ variables. However, the quadratic equations are the same set of equations as in the Intersection attack on UOV, so we know that they give only $\binom{k+1}{2}m - 2\binom{k}{2}$ linearly independent equations. We can then use the Bilinear XL variant of Smith-Tone and Perlner [19] to find the unique solution to the system of equations.

Remark 6. If we put $k = 1$ then we recover the Rainbow Band Separation attack (see Sect. 5.3), so our attack can be seen as a generalization of the RBS attack. However, note that previous works have assumed that only $n-1$ out of the n bilinear equations are useful. We find that this is not quite correct. Even though there is a syzygy at degree $(2, 1)$ (which we will discuss later) it is still useful to consider all n bilinear equations.

6.1 Extending to $n \geq 3o_2$

If $n \geq 3o_2$, then we expect there to be no non-trivial intersection, so the attack is not guaranteed to succeed with $k = 2$. However, if we model L_1O_2 and L_2O_2 as uniformly random subspaces of $O_2^{\perp}$, then the probability that they intersect non-trivially is approximately q^{-n+3o_2-1}. Therefore, we can expect the attack to succeed after q^{n-3o_2+1} guesses for (L_1, L_2).

6.2 Complexity analysis of the attack

The system of equations (6) is clearly not generic, since the first $\binom{k+1}{2}m$ equations only contain the entries of $\mathbf{x}$ as variables, and the remaining $k(n-k)$ equations are bi-linear in the entries of $\mathbf{x}$ and $\mathbf{v}$. This is the same structure as the systems that appear in the RBS attack (Sec. 5.3). Smith-Tone and Perlner investigated the complexity of solving such systems, and they proposed a variant of the XL algorithm that exploits the bi-homogeneous structure of the system [19]. Their algorithm works for systems of polynomial equations in

$n_x + n_y$ variables, where m_x equations are quadratic in the first n_x variables, and m_{xy} equations are bi-linear in the first n_x and last n_y variables respectively. Under a maximal rank assumption, their XL variant terminates at bi-degree (A, B) if the coefficient corresponding to $t^a s^b$ in

$$\frac{(1-t^2)^{m_x}(1-ts)^{m_{xy}}}{(1-t)^{n_x+1}(1-s)^{n_y+1}} \tag{7}$$

is non-positive for some a, b with $a \leq A$ and $b \leq B$. If this is the case, an upper bound for the number of multiplications in the attack is given by

$$3\overline{\mathcal{M}}(A,B)^2\binom{n_x+2}{2}, \tag{8}$$

where $\overline{\mathcal{M}}(A,B)$ is the number of monomials with bi-degree bounded by (A, B).

The maximal rank assumption is not valid for small instances of Rainbow, because there are k^2 non-trivial syzygies: For each $(i,j) \in \{1,\cdots,k\}^2$ we have

$$\begin{aligned}\langle \mathbf{v}, \mathcal{P}'(L_i^{-1}\mathbf{x}, L_j^{-1}\mathbf{x})\rangle &= \sum_{l=1}^{m} v_l \cdot \mathcal{P}'_l(L_i^{-1}\mathbf{x}, L_j^{-1}\mathbf{x}) \\ &= \sum_{t=1}^{m} \langle \mathbf{v}, \mathcal{P}'(L_i^{-1}\mathbf{x}, \mathbf{e}_t)\rangle \cdot (L_j^{-1}\mathbf{x})_t\,,\end{aligned}$$

which gives a non-trivial syzygy for the system (6) at bi-degree $(2, 1)$.

Since adding an equation with bi-degree (a, b) to the polynomial system corresponds to an extra factor $(1 - t^a s^b)$ in the generating function (7), it seems natural that a syzygy at degree (a, b) results in a factor $(1-t^a s^b)^{-1}$. We therefore conjecture that the generating function for the system (6) is

$$\frac{(1-t^2)^{m_x}(1-ts)^{m_{xy}}(1-t^2 s)^{-k^2}}{(1-t)^{n_x+1}(1-s)^{n_y+1}} \tag{9}$$

where

$$\begin{aligned} n_x &= \min(nk-(2k-1)o_2, n-1), & n_y &= o_2, \\ m_x &= \binom{k+1}{2}m - 2\binom{k}{2}, \quad \text{and} & m_{xy} &= kn\,. \end{aligned}$$

We experimentally verified that this generating function exactly predicts the ranks of the Macaulay matrices for small instances of Rainbow (see Table 4). That is, we found that the rank of the Macaulay matrix at bi-degree (A, B)

equals $\overline{\mathcal{M}}(A,B)$ minus the coefficient of $t^A s^B$ in (9), unless one of the coefficient of $t^a s^b$ with $a \leq A$ and $b \leq B$ is non-positive, in which case the rank is $\overline{\mathcal{M}}(A,B) - 1$, and the bilinear XL algorithm will succeed at bi-degree (A,B).

Under our assumption, we can estimate the cost of our attack by iterating over all minimal bi-degrees (A,B) for which the attack will succeed (i.e. for which the coefficient of $t^A s^B$ in the generating function is non-positive), and picking the bi-degree (A,B) that minimizes the cost (8).

6.3 Application to Rainbow NIST submissions

We now estimate the complexity of our attack on the Rainbow parameter sets that were submitted to the NIST PQC project. For all the proposed parameter sets we have $n \geq 3o_2$, which means the basic attack will need to be repeated multiple times before we expect to recover the secret key. For the Ia parameter set on the second-round submission, we have $n = 3m$, and for all the parameter sets of the final round submission we have $n = 3m + 4$. In these cases, we need to repeat the attack q and q^5 times respectively. For the IIIc and Vc parameter set of the second-round submission, n is much larger than $3m$, so the attack is very inefficient in these cases.

Table 3 reports the estimated gate count of our attack. To convert from the number of multiplications to the gate count, we use the model that is standard in the MQ literature; each multiplication costs $2(\log_2(q)^2 + \log_2(q))$ gates. We see that our attack outperforms the best known attacks for 4 out of the 6 proposed parameter sets. The improvement is the largest for the Ia parameter set of the first round and the Vc parameter set of the finals, where we improve on existing attacks by almost 20 bits.

Table 3. The estimated gate count of our Intersection attack on Rainbow compared to the best known attacks (taken from [19] for the second round parameters and the Rainbow NIST submission for the finals parameters).

Parameter set		Attack parameters n_x	n_y	m_x	m_{xy}	guesses	(A,B)	New attack	Known attacks
Second round	Ia	95	32	190	192	q^1	(10,1)	<u>123</u>	140
	IIIc	139	36	214	280	q^{33}	(6,9)	412	<u>204</u>
	Vc	187	48	286	376	q^{45}	(6,15)	548	<u>264</u>
Finals	Ia	99	32	190	200	q^5	(7,4)	<u>140</u>	147
	IIIc	147	48	238	296	q^5	(10,6)	<u>213</u>	217
	Vc	195	64	298	392	q^5	(10,12)	<u>262</u>	281

Table 4. The rank and the number of columns of the Macaulay matrices for the system of equations of the intersection attack. The rank at degree (A, B) always matches the coefficient of $t^A s^B$ in $\frac{1-(1-t^2)^{mx}(1-ts)^{mxy}(1-t^2s)^{-k^2}}{(1-t)^{nx}(1-s)^{ny}}$, except if the coefficient is larger or equal to the number of columns. In this case (marked by boldface in the table) the rank equals the number of columns minus 1, and the XL system can be solved at bi-degree (A, B).

parameters					Macaulay matrix at bi-degree (A,B)							
n	m	o_2	k		$(2,0)$	$(1,1)$	$(3,0)$	$(2,1)$	$(1,2)$	$(3,1)$	$(2,2)$	$(1,3)$
8	6	3	2	rank	16	12	**119**	**143**	64			**159**
				cols	36	32	120	144	80			160
10	6	3	1	rank	6	10	48	103	40	**479**	331	100
				cols	36	32	120	144	80	480	360	160
12	8	4	1	rank	8	12	72	147	60	795	589	180
				cols	45	45	165	225	135	825	675	315
12	8	4	2	rank	22	24	264	**389**	120			360
				cols	78	60	364	390	180			420
14	10	5	1	rank	10	14	100	199	84	1220	953	294
				cols	55	60	220	330	210	1320	1155	650
14	10	5	2	rank	28	28	392	556	168	**3359**	**2204**	588
				cols	105	84	560	630	294	3360	2205	784

7 The Rectangular MinRank Attack

In this section we introduce a new MinRank attack that exploits the property that for $\mathbf{y} \in O_2$, we have that $\mathcal{P}'(\mathbf{x}, \mathbf{y}) \in W$ for all $\mathbf{x} \in \mathbb{F}_q^n$. Let $\mathbf{e}_1, \cdots, \mathbf{e}_n$ be the basis for $\mathbb{F}_q^n$ where $\mathbf{e}_i$ is a vector whose entries are zero, except for the i-th entry which equals one. For a vector $\mathbf{x} \in \mathbb{F}_q^n$, we define the matrix

$$L_{\mathbf{x}} = \begin{pmatrix} \mathcal{P}'(\mathbf{e}_1, \mathbf{x}) \\ \cdots \\ \mathcal{P}'(\mathbf{e}_n, \mathbf{x}) \end{pmatrix} .$$

If $\mathbf{y} \in O_2$, then all the rows of $L_{\mathbf{y}}$ are in W, which implies that the matrix has rank at most $\dim W = o_2$. Moreover, it follows from the bilinearity of $\mathcal{P}'$ that

$$L_{\mathbf{y}} = \sum_{i=1}^{n} y_i L_{\mathbf{e}_1} .$$

Since the $L_{\mathbf{e}_i}$ matrices are public information, it follows that finding $\mathbf{y} \in O$ reduces to an instance of a rectangular MinRank problem; if an attacker can find a linear combination $\sum_{i=1}^{n} L_{\mathbf{e}_i} y_i$ with rank at most o_2, then we can assume that $\mathbf{y}$ is in O_2. If we set $o_2 - 1$ entries of $\mathbf{y}$ to zero, we still expect a non-trivial solution, so it suffices to look for a linear combinations of only the matrices $L_{\mathbf{e}_1}$ up to $L_{\mathbf{e}_{n-o_2+1}}$. Note that this MinRank instance is fundamentally different from the one that was already known in the literature (see Table 5).

Table 5. Comparison of the new MinRank instance with the known instance of the MinRank problem.

	Known instance of MinRank problem	New instance of MinRank problem
Size of matrices	n-by-n	n-by-m
Number of matrices	$o_2 + 1$	$n - o_2 + 1$
Rank of linear combination	m	o_2
Solution	vector in $W^{\perp}$	vector in O_2

We can use generic algorithms to solve this instance of the MinRank problem, such as the guessing strategy, or the algebraic methods of Sect. 5.3. However, in our case we can do slightly better because we have more information about $\mathbf{y}$; on top of knowing that $L_{\mathbf{y}}$ has low rank, we also know that $\mathcal{P}(\mathbf{y}) = 0$. Note that the variables y_i already appear in the system of equations that model the rank condition $\text{rank}(L_{\mathbf{y}}) \leq o_2$. Therefore, we can add the equations $\mathcal{P}(\mathbf{y}) = 0$ to the system without having to introduce additional variables. This will make the attack slightly more efficient.

7.1 Complexity Analysis

We first estimate the complexity of solving the pure MinRank problem with the support minors modeling approach of Sect. 2.3, without using the additional equations $\mathcal{P}(\mathbf{y}) = 0$. From experiments it seems that in case we are working in a field of odd characteristic, the MinRank instance behaves like a generic instance of the MinRank problem, so we can use the methodology of Bardet *et al.* to estimate the complexity of a random MinRank instance with $n - o_2 + 1$ matrices of size n-by-m with target rank o_2 (see Sect. 2.3). However, in case of a field with characteristic 2 (which includes all the Rainbow parameters submitted to NIST), there are some syzygies that do not appear in the case of random MinRank instances. This stems from the fact that, in characteristic 2, we have

$$\mathcal{P}'(\mathbf{y}, \mathbf{y}) = \mathcal{P}(2\mathbf{y}) - \mathcal{P}(\mathbf{y}) - \mathcal{P}(\mathbf{y}) = 2\mathcal{P}(\mathbf{y}) = 0\,,$$

so the $(r+1)$-by-$r+1$ minors of

$$\binom{\mathcal{P}'(\mathbf{y},\mathbf{y})}{C} = \sum_{i=0}^{n} y_i \binom{\mathcal{P}'(\mathbf{e}_i,\mathbf{y})}{C}$$

all vanish, which gives $\binom{m}{r+1}$ non-trivial linear relation between the equations at degree $(2,1)$. It is possible to carefully count how many linearly independent equations we have at each degree (b,i), with an analysis similar to the analysis of Bardet *et al.* [1].

However, to simplify the analysis, we can side-step the syzygies by ignoring one of the rows of the $L_1, \cdots, L_{n-o_2+1}$ matrices; since all the syzygies use all the rows of the L_i, the syzygies do not occur anymore if we omit a row from all the L_i matrices. Experimentally, we find that after removing a row, the instance behaves exactly like a random instance of the MinRank problem with $n-o_1+1$ matrices of size $(n-1)$-by-m and with rank o_2. We can therefore use the methodology of Bardet *et al.* to estimate the complexity of the attack (see Sect. 2.3). The first half of Table 6 reports on the estimated complexities for the Rainbow parameter sets that were submitted to the second round and the finals of the NIST PQC standardization project.

Table 6. The optimal attack parameters of the new MinRank attack, and the corresponding gate complexity for the Rainbow parameter sets submitted to the second round and the finals of the NIST PQC standardization project.

Parameter set		Plain MinRank			MinRank and $\mathcal{P}(\mathbf{y})=0$		
		m'	b	$\log_2$ gates	m'	b	$\log_2$ gates
Second round	Ia	51	2	131	40	6	124
	IIIc	59	2	153	52	4	151
	Vc	80	2	197	74	3	191
Finals	Ia	51	2	131	44	4	127
	IIIc	72	3	184	68	4	177
	Vc	95	4	235	87	6	226

The attack using $\mathcal{P}(\mathbf{y})=0$. We use the notation of Sect. 2.3, where M_b is the Macaulay matrix for the Support Minors Modelling system at bi-degree $(b,1)$ (omitting one row of the L_i matrices, as discussed earlier), and where $\mathcal{M}(b,1)$ is the number of monomials of degree b in the y_i variables and of degree 1 in the c_S variables. Let M_b^+ be the Macaulay matrix of the SMM system after appending the $\mathcal{P}(\mathbf{y}) = 0$ equations. We want to figure out the minimal value of b, for which the rank of M_b^+ is equal to $\mathcal{M}(b, 1) - 1$, because in that case the system $M_b^+\mathbf{x} = 0$ will have a one-dimensional solution space that corresponds to the solutions of the MinRank problem.

Bardet *et al.* already computed the rank of M_b, so we only need to figure out how much the rank increases by including the $\mathcal{P}(\mathbf{y}) = 0$ equations. Let $G(t)$ be a generating function for the dimension of the kernel of M_b, and $G^+(t)$ a generating function for the dimension of the kernel of M_b^+. Note that, even though we do not have a nice expression for $G(t)$, we can compute its coefficients from the expression of Bardet *et al.* for the rank of M_b, because the coefficient corresponding to t^b in $G(t)$ is $\mathcal{M}(b, 1) - \text{rank}(M_b)$. Under some genericity assumptions we have that $G^+(t) = (1 - t^2)^m G(t)$, from which we can get the rank of M_b^+.

Experimentally, we found for all the instances of Rainbow we could check, that this predicts the rank of M_b^+ exactly (see Table 7).

Table 7. The rank and the number of columns of the Macaulay matrices for the system of equations of the rectangular MinRank attack. The rank at bi-degree $(b, 1)$ always matches the predicted values, except if the prediction is larger or equal to the number of columns. In this case (marked by boldface in the table) the rank equals the number of columns minus 1, and the XL system can be solved at bi-degree $(b, 1)$.

parameters					Macaulay matrix at bi-degree $(b,1)$			
n	m	o_2	m'		$b=1$	$b=2$	$b=3$	$b=4$
				rank	40	244	**839**	
9	6	3	5	rank with $\mathcal{P}(\mathbf{y}) = 0$	40	**279**		
				number of columns	70	280	840	
				rank	66	528	2376	**7424**
12	8	4	6	rank with $\mathcal{P}(\mathbf{y}) = 0$	66	648	**2474**	
				number of columns	135	675	2475	7425
				rank	14	154	924	4004
15	10	5	6	rank with $\mathcal{P}(\mathbf{y}) = 0$	14	214	1444	**6005**
				number of columns	66	396	1716	6006
				rank	136	1615	10387	
18	12	6	8	rank with $\mathcal{P}(\mathbf{y}) = 0$	136	1951	**12739**	
				number of columns	364	2548	12740	

To estimate the complexity of the attack, we compute the first few terms of $G(t)$ until we encounter the first non-positive coefficient. If the first non-positive coefficient corresponds to t^b, then we assume the bilinear XL algorithm will work at bi-degree $(b, 1)$ and we can upper bound its cost as

$$3\mathcal{M}(b_{min}, 1)^2 W$$

multiplications, where $W = \max((o_2+1)(n-o_2+1), \binom{(n-o_2+3)}{2})$ is the maximal weight of the equations in the system. We found that, as already observed by Bardet *et al.* , it is helpful to consider only the first m' columns of the matrices $L_{\mathbf{e}_i}$. For each value of $m' \in [o_2+1, m]$ we estimate the attack cost, and we pick the value of m' that results in the smallest cost. The optimal attack parameters (m', b) and the corresponding costs (in terms of gate count) are reported in Table 6. We see that adding the $\mathcal{P}(\mathbf{y}) = 0$ equations to the Support minors modeling system reduces the attack complexity by a modest factor between 2^2 and 2^9 for the NIST parameter sets.

8 Conclusion

This paper offers a new perspective on the UOV and Rainbow signature schemes that avoids the use of a central map. This makes it easier to understand the existing attacks on these schemes, and allowed us to discover some new, more powerful, attacks. We hope that our simpler perspective will encourage more researchers to scrutinize the UOV and Rainbow signature schemes.

We introduce two new attacks: the intersection attack, which applies to both the UOV and the Rainbow signature schemes, and the rectangular MinRank Attack that applies only to the Rainbow scheme. Although methods for solving systems of multivariate quadratic equations (and our understanding of their complexity) have been improving over the last decades, the intersection attack is the first improvement in the cryptanalysis of UOV that is specific to the structure of the UOV public keys since 1999. Similarly, even though our understanding of the complexity of attacks on Rainbow has been improving (recent examples are [19] and [1]), there had not been any fundamentally new attacks on Rainbow since 2008.

New parameters for UOV and Rainbow. Both of our attacks reduce the security level of the Rainbow NIST submission below the requirements set out by NIST (see Table 8). However, our attacks are still exponential, and Rainbow can be saved by increasing the parameter sizes by a relatively small amount. For example, using $q = 16, n = 109, m = 68, o_2 = 36$ would presumably reach NIST security level I and would result in a signature size of 71 Bytes (a 10 %

Table 8. An overview of the estimated gate counts of our attacks versus known attacks and the target security level for the six Rainbow parameter sets submitted to the second round and the finals of the NIST PQC standardization project. The complexities of the known attacks are taken from [19] for the second round parameters and the Rainbow NIST submission for the finals parameters. The security target is taken from the NIST PQC call for proposals.

Parameter set		Intersection attack	New MinRank attack	Known attacks	Security target
Second round	Ia	$\underline{123}$	124	140	143
	IIIc	412	$\underline{151}$	204	207
	Vc	548	$\underline{191}$	264	272
Finals	Ia	140	$\underline{127}$	147	143
	IIIc	213	$\underline{177}$	217	207
	Vc	262	$\underline{226}$	281	272

increase) an key size of roughly 203 KB (an increase of 25 %). Alternatively, one could use the UOV scheme with $q = 64, n = 118, m = 47$, which results in 89 Byte signatures and a key size of 242 Kilobytes. It seems questionable whether the small performance advantage of Rainbow over UOV is worth the additional complexity. We leave a more carefully optimized parameter choice for UOV and Rainbow for future work.

Acknowledgments. I would like to thank Bo-Yin Yang and Jintai Ding for providing helpful feedback on an earlier version of this paper.

References

[1] Magali Bardet, Maxime Bros, Daniel Cabarcas, Philippe Gaborit, Ray Perlner, Daniel Smith-Tone, Jean-Pierre Tillich, and Javier Verbel. Improvements of algebraic attacks for solving the rank decoding and MinRank problems. In *International Conference on the Theory and Application of Cryptology and Information Security*, 2020.

[2] Magali Bardet, Jean-Charles Faugere, Bruno Salvy, and Bo-Yin Yang. Asymptotic behaviour of the degree of regularity of semi-regular polynomial systems. In *Proc. of MEGA, Eighth International Symposium on Effective Methods in Algebraic Geometry*, volume 5, 2005.

[3] Ward Beullens, Bart Preneel, Alan Szepieniec, and Frederik Vercauteren. LUOV. Technical report, National Institute of Standards and Technology, 2019. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions`.

[4] Olivier Billet and Henri Gilbert. Cryptanalysis of Rainbow. In Roberto De Prisco and Moti Yung, editors, *SCN 06*, volume 4116 of *LNCS*, pages 336–347. Springer, Heidelberg, September 2006.

[5] Antoine Casanova, Jean-Charles Faugère, Gilles Macario-Rat, Jaques Patarin, Ludovic Perret, and Jocelyn Ryckeghem. GeMSS. Technical report, National Institute of Standards and Technology, 2019. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions`.

[6] Nicolas Courtois, Alexander Klimov, Jacques Patarin, and Adi Shamir. Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 392–407. Springer, Heidelberg, May 2000.

[7] Peter Czypek, Stefan Heyse, and Enrico Thomae. Efficient implementations of MQPKS on constrained devices. In Emmanuel Prouff and Patrick Schaumont, editors, *CHES 2012*, volume 7428 of *LNCS*, pages 374–389. Springer, Heidelberg, September 2012.

[8] Jintai Ding, Ming-Shing Chen, Albrecht Petzoldt, Dieter Schmidt, and Bo-Yin Yang. Rainbow. Technical report, National Institute of Standards and Technology, 2019. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions`.

[9] Jintai Ding and Dieter Schmidt. Rainbow, a new multivariable polynomial signature scheme. In *International Conference on Applied Cryptography and Network Security*, pages 164–175. Springer, 2005.

[10] Jintai Ding, Bo-Yin Yang, Chia-Hsin Owen Chen, Ming-Shing Chen, and Chen-Mou Cheng. New differential-algebraic attacks and reparametrization of Rainbow. In Steven M. Bellovin, Rosario Gennaro, Angelos D. Keromytis, and Moti Yung, editors, *ACNS 08*, volume 5037 of *LNCS*, pages 242–257. Springer, Heidelberg, June 2008.

[11] Jean Charles Faugère. A new efficient algorithm for computing gröbner bases without reduction to zero (F_5). In *Proceedings of the 2002 international symposium on Symbolic and algebraic computation*, pages 75–83, 2002.

[12] Jean-Charles Faugère, Mohab Safey El Din, and Pierre-Jean Spaenlehauer. Computing loci of rank defects of linear matrices using gröbner bases and applications to cryptology. In *Proceedings of the 2010 International Symposium on Symbolic and Algebraic Computation*, pages 257–264, 2010.

[13] Louis Goubin and Nicolas Courtois. Cryptanalysis of the TTM cryptosystem. In Tatsuaki Okamoto, editor, *ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 44–57. Springer, Heidelberg, December 2000.

[14] Aviad Kipnis, Jacques Patarin, and Louis Goubin. Unbalanced oil and vinegar signature schemes. In Jacques Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 206–222. Springer, Heidelberg, May 1999.

[15] Aviad Kipnis and Adi Shamir. Cryptanalysis of the oil & vinegar signature scheme. In Hugo Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 257–266. Springer, Heidelberg, August 1998.

[16] Aviad Kipnis and Adi Shamir. Cryptanalysis of the HFE public key cryptosystem by relinearization. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 19–30. Springer, Heidelberg, August 1999.

[17] Jacques Patarin. The oil and vinegar signature scheme. In *Dagstuhl Workshop on Cryptography September, 1997*, 1997.

[18] Simona Samardjiska, Ming-Shing Chen, Andreas Hülsing, Joost Rijneveld, and Peter Schwabe. MQDSS. Technical report, National Institute of Standards and Technology, 2019. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions`.

[19] Daniel Smith-Tone and Ray Perlner. Rainbow band separation is better than we thought. Technical report, Cryptology ePrint Archive preprint, 2020.

Chapter 12

CSI-FiSh

—–BEGIN PGP SIGNED MESSAGE—–
Hash: SHA1

Any chance you can stop the job running on my machine (Lumagon)? It is stopping me working.

Nigel Smart
COSIC - KU Leuven

– Nigel Smart, personal communication, 3/27/2019

Publication Data

Beullens W., Kleinjung T., Vercauteren F. CSI-FiSh: Efficient Isogeny Based Signatures Through Class Group Computations. In: Advances in Cryptology – ASIACRYPT 2019. Lecture Notes in Computer Science, vol 11921, pages 227-247, Springer, 2019.

My Contribution

Frederik and I wrote the paper and figured out the details of the protocol together, although most of the design was already known by Rostovstev-Stolbunov, and described in the SeaSign paper. Thorsten provided the implementation of the algorithms for computing the class group. Managing the class group computation (scheduling and monitoring compute jobs on ESAT machines, processing results, dealing with angry colleagues whose machines were running slow, etc) was mostly my responsibility. I wrote the implementation of CSI-FiSh and did the benchmarking.

CSI-FiSh: Efficient Isogeny based Signatures through Class Group Computations

Ward Beullens[1], Thorsten Kleinjung[2], and Frederik Vercauteren[1]

`ward.beullens@esat.kuleuven.be`, `thorsten.kleinjung@epfl.ch`, `frederik.vercauteren@esat.kuleuven.be`

[1] imec-COSIC, ESAT, KU Leuven, Belgium
[2] EPFL IC LACAL, Station 14, CH-1015 Lausanne, Switzerland

Abstract. In this paper we report on a new record class group computation of an imaginary quadratic field having 154-digit discriminant, surpassing the previous record of 130 digits. This class group is central to the CSIDH-512 isogeny based cryptosystem, and knowing the class group structure and relation lattice implies efficient uniform sampling and a canonical representation of its elements. Both operations were impossible before and allow us to instantiate an isogeny based signature scheme first sketched by Stolbunov. We further optimize the scheme using multiple public keys and Merkle trees, following an idea by De Feo and Galbraith. We also show that including quadratic twists allows to cut the public key size in half for free. Optimizing for signature size, our implementation takes 390ms to sign/verify and results in signatures of 263 bytes, at the expense of a large public key. This is 300 times faster and over 3 times smaller than an optimized version of SeaSign for the same parameter set. Optimizing for public key and signature size combined, results in a total size of 1468 bytes, which is smaller than any other post-quantum signature scheme at the 128-bit security level.

Keywords: Isogeny based cryptography, digital signature, class group, group action, Fiat-Shamir.

1 Introduction

Isogeny based cryptography was first proposed in 1997 by Couveignes [9] in a talk at the “séminaire de complexité et cryptographie” at the ENS, but

[*] This work was supported in part by the Research Council KU Leuven grants C14/18/067 and STG/17/019. Ward Beullens is funded by an FWO fellowship. Date of this document: 2019.05.14.

his ideas on how class group actions could be used in cryptography were not published at that time. The same ideas were independently rediscovered in 2006 by Rostovtsev and Stolbunov [31]. Both Couveignes as well as Rostovtsev and Stolbunov described a Diffie-Hellman like key agreement scheme (usually called CRS) using the class group of the endomorphism ring of ordinary elliptic curves. Rostovtsev and Stolbunov also describe an isogeny based identification scheme. However, none of these schemes can be considered practical.

A different approach was taken by Jao and De Feo who introduced SIDH (Supersingular Isogeny Diffie–Hellman) [22]. SIDH does not rely on class group actions as CRS, but exploits the simple fact that dividing out an elliptic curve by two (large) non-intersecting subgroups is commutative. SIDH uses supersingular curves, mainly for two reasons: firstly, constructing a supersingular elliptic curve with given group order is trivial, and secondly, their endomorphism ring is non-commutative which thwarts attacks by Kuperberg's algorithm [25]. SIDH forms the basis of a practical key-exchange protocol called SIKE [21], which is one of the main contenders in NIST's post-quantum standardization project [29].

A major improvement of CRS was made by Castryck et. al. [6] by instantiating the scheme for supersingular curves over $\mathbb{F}_p$ and by restricting the endomorphism ring to $\mathbb{F}_p$-rational endomorphisms. This subring behaves very much like in the ordinary curve setting, so the CRS approach applies. The main advantage is that the class group action can be computed very efficiently since by construction, the supersingular curves have many small rational subgroups. The resulting cryptosystem is called CSIDH for Commutative Supersingular Isogeny Diffie-Hellman and is pronounced "sea-side".

Both SIDH and CSIDH result in efficient key-agreement schemes, but a practical isogeny based signature scheme is much harder to achieve. The first attempt was made by Stolbunov in his PhD thesis [35]; the signature scheme consists of the Fiat-Shamir transform applied to a standard three pass isogeny based identification scheme. The scheme can be securely instantiated under two assumptions: firstly, it should be possible to sample uniformly in the class group (this could be efficiently approximated) and secondly, each element in the class group has an efficiently computable canonical representation. Especially the second assumption is a major obstacle to instantiate Stolbunov's signature scheme.

This problem was partly remedied by De Feo and Galbraith in the signature scheme SeaSign [11] by employing "Fiat–Shamir with aborts". The main idea is, instead of using a canonical representation for each class group element, to use a majorly redundant representation and to apply rejection sampling to make the distribution of the class group elements, which are part of the signature, independent of the secret key. The drawback is that this redundant representation makes evaluating the class group much less efficient. Several versions of SeaSign were presented offering trade-offs between signature size, public-key size, and secret-key size. Although signature sizes of less than one kilobyte at the 128-bit security level are possible, the scheme is again not practical taking several minutes to sign. Decru et al. [12] improved all variants of SeaSign, but the fastest parameter set still requires 2 minutes to sign a message.

A different approach was taken by Yoo et al. [38] who transform an SIDH-based zero-knowledge proof proposed by De Feo et al. [15] into a digital signature scheme. The resulting signatures however are rather large at $\sim$ 120KB which is much larger than other post-quantum signature schemes. A similar approach was described by Galbraith et al. [17] who were able to compress the signatures down to roughly 10KB. None of the above signature schemes is therefore practical, either due to lack of efficiency or due to the large signatures.

It is well known (see for instance Couveignes [9], Stolbunov's PhD [35] or Section 9.2 of [11]), that knowing the class group structure would resolve the two main problems with Stolbunov's signature scheme. Firstly, uniform sampling is now trivial, but more importantly, each element has an efficiently computable canonical representation. This immediately implies that rejection sampling is no longer necessary, thereby majorly speeding up the resulting signature scheme.

The computation of the class group of a quadratic imaginary number field is a classical problem in computational number theory, and the current best algorithms [20, 4, 23] are improvements of an algorithm due to Hafner and McCurley [18]. These algorithms have complexity $L_{1/2}(\Delta)$ with Δ the discriminant of the number field. The largest publicly known class group computation was for a 130-digit discriminant by Kleinjung [23].

The main contributions in this paper are as follows:

- We compute the class group structure and a relation lattice of the class group of the quadratic imaginary field corresponding to the CSIDH-512 parameter set having a 154-digit discriminant. This computation is described in Section 3.
- We present an efficient algorithm to compute the class group action of random class group elements by solving an approximate CVP-problem in the relation lattice. This strategy is described in Section 4 and is a combination of Babai nearest plane algorithm [1] and a random walk approach due

to Doulgerakis, Laarhoven and de Weger [14]. Compared to native CSIDH which starts from an efficient representation, our algorithm is only 15% slower.
- In Section 5, we introduce CSI-FiSh (Commutative Supersingular Isogeny based Fiat-Shamir signatures, pronounce "sea-fish") which is based on Stolbunov's signature scheme [35] combined with optimisations similar to the ones described for SeaSign [11]. We also show that the public key size can be cut in half for free by including not only the curve, but also its quadratic twist. This implicitly doubles the number of curves in the public key for free, without affecting the security of the scheme. Finally, we prove that the resulting signature scheme is secure in the quantum random oracle model.
- We provide an efficient open-source implementation of CSI-FiSh and report on the implementation results in Section 6. As for SeaSign, CSI-FiSh allows for various trade-offs: the smallest signatures are 263 bytes and are also the fastest ($\sim$ 390ms to sign/verify), but require a large public key of 2 MB. Slightly larger signatures of 461 bytes require a public key of 16KB which is comparable to multivariate schemes such as LUOV [3], but take $\sim$ 670ms to compute. Optimizing for public key and signature size combined, results in a total size of 1468 bytes which is smaller than any other post-quantum signature scheme at the 128-bit security level.

2 Preliminaries

We denote by $[a, b]$ with $a, b \in \mathbb{Z}, a \leq b$ the set $\{a, \ldots, b\}$. When considering reals instead of integers $[a, b]$ denotes the interval $a \leq r \leq b$ with $r \in \mathbb{R}$, whereas $[a, b[$ denotes $a \leq r < b$. The cardinality of a set S is denoted by $\#S$.

2.1 Elliptic curves and isogenies

The go-to general reference on elliptic curves is Silverman [33]. A good introduction to isogeny based cryptography can be found in the lecture notes by De Feo [10].

Let E be an elliptic curve over a finite field $\mathbb{F}_p$ with p a large prime, and let $\mathbf{0}_E$ denote the point at infinity on E. The curve E is called supersingular iff $\#E(\mathbb{F}_p) = p + 1$, and ordinary otherwise. Given two elliptic curves E and E', an isogeny ϕ is a morphism $\phi : E \to E'$ (i.e. can be expressed as fractions of polynomials) such that $\phi(\mathbf{0}_E) = \mathbf{0}_{E'}$. An isomorphism is an isogeny that has an inverse (which is also a morphism), and two elliptic curves are isomorphic iff they have the same j-invariant, which is a simple algebraic expression in the coefficients of the curve. Since an isogeny defines a group homomorphism from E to E', its kernel is a subgroup of E. Vice-versa, any subgroup $S \subset E(\mathbb{F}_{p^k})$ determines a (separable) isogeny $\phi : E \to E'$ with $\ker \phi = S$, i.e. $E' = E/S$. The equation for E' and the isogeny ϕ can be computed using Vélu's formulae [36] using $O(\#S(k \log p)^2)$ bit-operations. As such, it is only practical to handle fairly small subgroups S defined over small extensions of $\mathbb{F}_p$.

The ring of endomorphisms $\mathrm{End}(E)$ consists of all isogenies from E to itself, and $\mathrm{End}_{\mathbb{F}_p}(E)$ denotes the ring of endomorphisms defined over $\mathbb{F}_p$. For an ordinary curve $E/\mathbb{F}_p$ we have $\mathrm{End}(E) = \mathrm{End}_{\mathbb{F}_p}(E)$, but for a supersingular curve over $\mathbb{F}_p$ we have a strict inclusion $\mathrm{End}_{\mathbb{F}_p}(E) \subsetneq \mathrm{End}(E)$. In particular, it is known that for a supersingular curve over $\mathbb{F}_p$ its full endomorphism ring $\mathrm{End}(E)$ is an order in a quaternion algebra, whereas $\mathrm{End}_{\mathbb{F}_p}(E)$ is only an order in the imaginary quadratic field $\mathbb{Q}(\sqrt{-p})$. In the following we will denote this order $\mathcal{O} = \mathrm{End}_{\mathbb{F}_p}(E)$.

The ideal class group of $\mathcal{O}$ is the quotient of the group of fractional invertible ideals in $\mathcal{O}$ by the principal fractional invertible ideals, and will be denoted $\mathrm{Cl}(\mathcal{O})$. Given an $\mathcal{O}$-ideal $\mathfrak{a}$, we can consider the subgroup defined by the intersection of the kernels of the endomorphisms in $\mathfrak{a}$, i.e. $S_\mathfrak{a} = \bigcap_{\alpha \in \mathfrak{a}} \ker \alpha$. Since this is a subgroup of E, we can divide out by $S_\mathfrak{a}$ and denote the isogenous curve $E/S_\mathfrak{a}$ by $\mathfrak{a} \star E$. This isogeny is well-defined and unique up to $\mathbb{F}_p$-isomorphism and the group $\mathrm{Cl}(\mathcal{O})$ acts via the operator $\star$ on the set $\mathcal{E}$ of $\mathbb{F}_p$-isomorphism classes of elliptic curves with $\mathbb{F}_p$-rational endomorphism ring $\mathcal{O}$. One can show that $\mathrm{Cl}(\mathcal{O})$ acts freely and transitively on $\mathcal{E}$, i.e. $\mathcal{E}$ is a principal homogeneous space for $\mathrm{Cl}(\mathcal{O})$.

In what follows we will assume that the class group $\mathrm{Cl}(\mathcal{O})$ is cyclic of order $N = \#\mathrm{Cl}(\mathcal{O})$ generated by the class of an ideal $\mathfrak{g}$. The more general case of non-cyclic class groups is a trivial extension and is not required in the application we consider.

2.2 CSIDH

Castryck et al. [6] proposed an efficient commutative group action $\star$ by crafting supersingular elliptic curves with many small $\mathbb{F}_p$-rational subgroups. Given that $\#E(\mathbb{F}_p) = p + 1$ for a supersingular curve, it is immediate that if p is chosen to be of the form $4 \cdot \ell_1 \cdots \ell_n - 1$, with ℓ_i small distinct odd primes, we have

$\#E(\mathbb{F}_p) = 4 \cdot \ell_1 \cdots \ell_n$. Such curves therefore have, for each $i \in [1, n]$, an $\mathbb{F}_p$-rational subgroup of order ℓ_i. Since $p = -1 \bmod \ell_i$, we have that in $\mathbb{Q}(\sqrt{-p})$ the rational prime ℓ_i splits as $(\ell_i) = \langle \ell_i, \pi - 1\rangle\langle \ell_i, \pi + 1\rangle$, where $\pi = \sqrt{-p}$ represents the $\mathbb{F}_p$-Frobenius endomorphism. Note that the first ideal factor $\mathfrak{l}_i = \langle \ell_i, \pi - 1\rangle$ corresponds to the subgroup of order ℓ_i defined over $\mathbb{F}_p$, and that the action of this ideal can be computed entirely over $\mathbb{F}_p$. Once this subgroup is determined, Vélu's formulae require $O(\ell_i(\log p)^2)$ bit operations. However, for small ℓ_i, finding a generator of this small subgroup requires (at least one) full-size scalar multiplication which dominates the cost of Vélu's formulae.

CSIDH considers the action of ideals of the form $\prod_{i=1}^n \mathfrak{l}_i^{e_i}$ where the exponents are chosen uniformly from some interval $[-B, B]$. This can be done by computing sequentially the action of $\mathfrak{l}_i$ exactly e_i times. Since the cost of each such action is dominated by the cost to determine the correct subgroup, we assume that the overall cost of computing such action is mostly determined by the ℓ_1-norm of its exponent vector, i.e. $|e_1| + \cdots + |e_n|$.

The base curve is taken to be $E_0 \colon y^2 = x^3 + x$ over $\mathbb{F}_p$ and instead of using the j-invariant, each isomorphism class of a curve with given endomorphism ring $\mathrm{End}_{\mathbb{F}_p}(E) = \mathcal{O} = \mathbb{Z}[\pi]$ is represented by a single coefficient $A \in \mathbb{F}_p$ defining the curve $E_A \colon y^2 = x^3 + Ax^2 + x$. Denote $\mathcal{A}$ the set of all such coefficients A, then we obtain a class group action $\star : \mathrm{Cl}(\mathcal{O}) \times \mathcal{A} \to \mathcal{A}$ or equivalently, assuming the class group is cyclic of order N, a group action $[] : \mathbb{Z}_N \times \mathcal{A} \to \mathcal{A}$. To simplify notation in the remainder of the paper, we will identify a curve E_A with its isomorphism class represented by the corresponding coefficient A.

Note however that in CSIDH, the order (and structure) of the class group are unknown, so only the action of ideals of the form $\prod_{i=1}^n \mathfrak{l}_i^{e_i}$ with e_i smallish are computable. This restriction brings up various questions: firstly, given the range of exponent vectors $[-B, B]^n$, do the ideals $\prod_{i=1}^n \mathfrak{l}_i^{e_i}$ cover the whole class group, and secondly, assuming the exponents are chosen uniformly in $[-B, B]$, is the resulting distribution of $\prod_{i=1}^n \mathfrak{l}_i^{e_i}$ uniform over $\mathrm{Cl}(\mathcal{O})$. It is clear that knowing the class group structure voids both questions as surjectivity and uniformity become trivial to attain. The only remaining problem then is to efficiently compute the action $[a]$ given a random exponent $a \in \mathbb{Z}_N$ (see Section 4 for an efficient solution).

2.3 Computational problems

The main hardness assumption underlying group actions based on isogenies, is that it is hard to invert the group action:

Definition 1 (Group Action Inverse Problem (GAIP)). *Given a curve E, with $\mathrm{End}(E) = \mathcal{O}$, find an ideal $\mathfrak{a} \subset \mathcal{O}$ such that $E = \mathfrak{a} \star E_0$.*

Another advantage of knowing the class group structure and therefore uniform sampling, is that the GAIP is random self-reducible: given a problem instance E, we can shift this over a uniformly random $\mathfrak{b}$ to obtain $E' = \mathfrak{b} \star E$, which is uniformly distributed in $\mathcal{A}$. Given a solution $\mathfrak{c}$ for E', it is easy to see that $\mathfrak{c}\mathfrak{b}^{-1}$ is then a solution to the original problem.

The CSI-FiSh signature scheme relies on the hardness of random instances of a multi-target version of the inversion problem, which is shown to reduce tightly to the normal GAIP by [11] in the case that the class group structure is known.

Definition 2 (Multi-Target GAIP (MT-GAIP)). *Given k elliptic curves $E_1, \ldots, E_k$ with $\mathrm{End}(E_1) = \cdots = \mathrm{End}(E_k) = \mathcal{O}$, find an ideal $\mathfrak{a} \subset \mathcal{O}$ such that $E_i = \mathfrak{a} \star E_j$ for some $i, j \in \{0, \ldots, k\}$ with $i \neq j$.*

The best classical algorithm to solve the GAIP problem is a simple meet-in-the-middle approach, where one finds a collision between two breadth-first trees starting at E and E' respectively. The time complexity of this approach is $O(\sqrt{\#\mathrm{Cl}(\mathcal{O})})$. The best quantum algorithm for the GAIP problem reformulates it as a hidden shift problem [7] and then applies Kuperberg's algorithm [25, 26], which runs in time $2^{O(\sqrt{\log N})}$. Translating this subexponential complexity to concrete security estimates is a highly non-trivial endeavour and we refer to [6, Section 7] for precise details.

In this paper we will only focus on the CSIDH-512 parameter set, which uses 74 small primes ℓ_i (so $n = 74$) and samples the exponents uniformly from the interval $[-5, 5]$ (so $B = 5$). The CSIDH authors assume that sampling exponent vectors in $[-5, 5]$ covers a subset of size $\sim 2^{256}$, which, as we will see, is a bit less than half of the total size of the class group. Class group elements (represented by their exponent vectors) require roughly 32 bytes, and each isomorphism class requires 64 bytes (one coefficient in $\mathbb{F}_p$). The average time taken to perform one such group action [6] is roughly 40 ms on a 3.5GHz processor. This parameter set aimed to provide 128-bit classical security and to achieve NIST security level 1 quantumly [6]. However, recent works propose quantum attacks that are claimed to break the NIST security level 1 claim [30, 5].

3 Class group computation

In order to uniformly sample and canonically represent class group elements, a class group computation of Hafner-McCurley type [18] was performed which, besides computing generators of the class group, also expresses the ideal classes of prime ideals with small norm in terms of these generators. This computation relied on the programs from [23], which work over the maximal order and thus we obtain generators for $\mathrm{Cl}(\mathcal{O}_{\mathbb{Q}(\sqrt{-p})})$, where p is the 512-bit prime used in CSIDH-512. This class group turns out to be cyclic and the class number is not divisible by 3. Since the conductor of the suborder $\mathcal{O}$ is (2) and 2 does not split in $\mathcal{O}_{\mathbb{Q}(\sqrt{-p})}$, we get $\#\mathrm{Cl}(\mathcal{O}) = 3\#\mathrm{Cl}(\mathcal{O}_{\mathbb{Q}(\sqrt{-p})})$ so that $\mathrm{Cl}(\mathcal{O})$ is also cyclic. Using the information from the computation over the maximal order, it is easy to find a generator of $\mathrm{Cl}(\mathcal{O})$ and to express the $\mathfrak{l}_i$ as powers of this generator. In total, the computation took an estimated effort of 52 core years on an inhomogenous cluster of number crunchers and desktop machines, consisting of around 800 cores with the "average" core running at around 3.3GHz.

The class group computation consists of the following steps.

Relation collection. Given a bound F (we chose $F = 7000000$), let $\mathcal{F}$ be the set of prime ideals of degree one with norm less than F and the prime ideal (2); the latter is only included for technical reasons. A relation is a decomposition $(a + \sqrt{-p}) = \prod_{\mathfrak{p}\in\mathcal{F}} \mathfrak{p}^{e_{a,\mathfrak{p}}}$ with $a, e_{a,\mathfrak{p}} \in \mathbb{Z}$. Such relations can be found by factoring the ideal $(a + \sqrt{-p})$ for random $a \in \mathbb{Z}$ which essentially amounts to factoring its norm $a^2 + p$. Since most a do not give rise to a relation, there exist many methods to speed up the search for relations. We used a sieving approach [23] and the large prime variation with up to three large primes; these details do not matter in the following and are suppressed.

The goal of this step is to generate sufficiently many relations such that the subsequent steps are able to determine the class group. In practice, this usually means that we can stop collecting relations when the number of relations slightly exceeds the number of prime ideals contained in their decompositions (which is at most $\#\mathcal{F}$). However, a bigger excess often reduces the running time of the subsequent steps significantly.

This step is one of the two main steps in terms of computational effort. Fortunately, it is trivially parallelized and has moderate memory requirements. In our computation it took an estimated time of 43 core years to collect 319.5 million relations over an extended factor base of size 32.7 million.

Building the matrix. In this step the set of relations is converted into a matrix over $\mathbb{Z}$ with rows corresponding to prime ideals and columns corresponding to relations; the matrix entry belonging to the prime ideal $\mathfrak{p} \in \mathcal{F}$ and relation $(a+\sqrt{-p})$ is $e_{a,\mathfrak{p}}$. This matrix is overdetermined and very sparse. We now assume that the ideal classes of the prime ideals in $\mathcal{F}$ generate the class group. In practice, it is very likely that this assumption holds; moreover, it follows from GRH if $\mathcal{F}$ is chosen appropriately. Under the assumption above, one has a surjection $\mathbb{Z}^{\#\mathcal{F}}/\Lambda \to \mathrm{Cl}(\mathcal{O}_{\mathbb{Q}(\sqrt{-p})})$ where Λ is the lattice spanned by the columns. If the matrix has full rank, the covolume of Λ is a multiple of the class number. By performing elementary column operations as well as removing certain rows and columns one can reduce this matrix significantly while keeping it slightly overdetermined and sparse; this is done to reduce the complexity of the next steps.

In terms of running time this step is negligible but it has higher memory requirements and is not easily parallelisable. We reduced our set of 319.5 million relations over a factor base of size 32.7 million to a slightly overdetermined matrix with roughly 222 thousand rows.

Matrix step. By dropping some columns from the matrix above one can obtain a square matrix and use the (block) Wiedemann algorithm modulo many small primes to compute its determinant over $\mathbb{Z}$ (cf. [37, 8]). If the determinant is non-zero, it is a (usually) huge multiple of the class number. By repeating the determinant calculation for another square matrix obtained by dropping another set of columns one gets a second huge multiple of the class number. Their greatest common divisor is much smaller, thus can be factored, and for each of its prime factors one can check whether it is a divisor of the class number using quadratic forms.

This is the other main step, it is also easy to parallelize and has moderate memory requirements. For both determinant computations, we computed the determinant modulo roughly 7000 different 64-bit primes, which took roughly 4.3 core years per determinant. By taking the gcd of the determinants and removing an extra factor of 2, we obtained that

$$\begin{aligned}\#\mathrm{Cl}(\mathcal{O}_{\mathbb{Q}(\sqrt{-p})}) = 37 \times 1407181 \times 51593604295295867744293584889\\ \times 31599414504681995853008278745587832204909\,.\end{aligned}$$

The class group of the order $\mathcal{O}$ therefore has cardinality $3 \cdot \#\mathrm{Cl}(\mathcal{O}_{\mathbb{Q}(\sqrt{-p})})$ which is approximately equal to $2^{257.136}$.

Final computations. In this step the r-Sylow group of $\mathrm{Cl}(\mathcal{O}_{\mathbb{Q}(\sqrt{-p})})$ is computed for each r dividing the class number together with the images of all involved prime ideals in this Sylow group. For small r this is easy and for large r the kernel of one of the square matrices from the previous step can be computed modulo r, e.g., using the Lanczos or Wiedemann algorithm. Finally, tying everything together a set of generators of the class group and for each involved prime ideal a representation in terms of these generators are obtained.

This step is negligible in terms of running time and has only moderate memory requirements. It turns out that the ideal $\mathfrak{l}_1 = \langle 3, \pi - 1\rangle$ generates $\mathrm{Cl}(\mathcal{O})$, the discrete logs of the other $\mathfrak{l}_i$ are available in our GitHub repository [2].

Remark 3. Notice that all odd primes up to 373 split in $\mathbb{Q}(\sqrt{-p})$ thus improving the probablity that the ideal $(a + \sqrt{-p})$ gives rise to a relation. This facilitates the class group computation for our choice of p but the gain is much less than a factor of 2 compared to an average prime of the size of p.

4 Class group action

In this section we discuss how to compute the action of ideals represented as $\mathfrak{g}^a$, where $\mathfrak{g}$ is a generator of the class group. In practice, it will often be the case that one of the $\mathfrak{l}_i$ generates the class group already, and in fact, for the CSIDH-512 class group we can even take $\mathfrak{g} = \mathfrak{l}_1 = \langle 3, \pi - 1\rangle$. Recall that for isogenies, there is no analogue of the standard square-and-multiply for exponentiation, so a different approach is required. We can only compute the group action efficiently for the prime ideals $\mathfrak{l}_i = \langle l_i, \pi - 1\rangle$, our approach is to first use lattice reduction algorithms to rewrite $\mathfrak{g}^a$ as a product of the $\mathfrak{l}_i$ with small exponents. After this step, the action can be computed efficiently with Vélu's Formulae.

Concretely, the ideal $\mathfrak{l}_1^a$ corresponds to the exponent vector $\mathbf{e} = [a, 0, \ldots, 0]$, that needs to be reduced modulo the relation lattice:

$$L := \{\mathbf{z} = (z_1, \ldots, z_n) \in \mathbb{Z}^n : \prod_{i=1}^{n} \mathfrak{l}_i^{z_i} = (1)\}\,.$$

The lattice L has rank n and volume $N = \#\mathrm{Cl}(\mathcal{O})$ since by definition it is the kernel of the surjective group homomorphism that maps $\mathbb{Z}^n \to \mathrm{Cl}(\mathcal{O}) : \mathbf{z} = (z_1, \ldots, z_n) \mapsto \prod_{i=1}^{n} \mathfrak{l}_i^{z_i}$. Note that the relation lattice follows directly from the class group computation described in Section 3.

Since the complexity of a CSIDH action is mainly determined by the ℓ_1-norm of the exponent vector, we want to solve the Closest Vector Problem (CVP) in L for the ℓ_1-norm given the target vector $\mathbf{e}$. Indeed, any vector $\mathbf{z} \in L$ which is close to $\mathbf{e}$ for the ℓ_1 norm will result in an equivalent vector $\mathbf{e} - \mathbf{z}$ such that $\|\mathbf{e} - \mathbf{z}\|_1$ is small and thus efficiently computable.

A first approximation for solving the CVP for the ℓ_1-norm is to use either Babai's rounding or nearest plane algorithm [1]. Given a set of basis vectors $B := \{\mathbf{b}_1, \dots, \mathbf{b}_n\}$, denote with $B^\star := \{\mathbf{b}_1^\star, \dots, \mathbf{b}_n^\star\}$ the corresponding Gram-Schmidt orthogonalization vectors. Let $\mathcal{P}(B)$ denote the parallelepiped

$$\mathcal{P}(B) = \left\{ \sum_{i=1}^{n} \alpha_i \mathbf{b}_i \mid \alpha_i \in [-1/2, 1/2[\right\} ,$$

then Babai rounding returns a lattice vector in $\mathbf{e} + \mathcal{P}(B)$ and Babai's nearest plane in $\mathbf{e} + \mathcal{P}(B^\star)$. This shows that $\mathbf{e} - \mathbf{z}$ is either in $\mathcal{P}(B)$ or in $\mathcal{P}(B^\star)$ depending on the choice of algorithm. Given a basis B and corresponding Gram-Schmidt basis $B^\star$, it is therefore easy to bound $\|\mathbf{e} - \mathbf{z}\|_1$. This also shows that a basis with short and almost orthogonal vectors will give better results. In our experiments, we only used Babai's nearest plane algorithm since it is superior to Babai rounding.

Several notions of lattice reduction (and corresponding reduction algorithms) exist such as LLL [28], BKZ [32] or HKZ [24]. Since the lattice L is fixed for a given class group, a considerable effort can be spent in reducing the lattice basis during a precomputation. To analyze the impact of the quality of the basis, we computed three reductions: BKZ-40, BKZ-50 and HKZ. For each reduced basis, we then ran Babai nearest plane resulting in Table 1, where the average ℓ_1-norm and standard deviation are given for a sample size of 10^4 random exponents.

Table 1. ℓ_1-norm and ℓ_2-norm of Babai's nearest plane method and evaluation times of CSIDH-action on three different bases

	BKZ-40	BKZ-50	HKZ
ℓ_1-norm	$\mu = 240.67$ $\sigma = 18.82$	$\mu = 239.35$ $\sigma = 18.35$	$\mu = 237.50$ $\sigma = 18.26$
ℓ_2-norm	$\mu = 35.13$ $\sigma = 2.47$	$\mu = 34.93$ $\sigma = 2.43$	$\mu = 34.67$ $\sigma = 2.38$
action evaluation time (10^6 cycles)	$\mu = 148.59$ $\sigma = 12.91$	$\mu = 148.41$ $\sigma = 12.57$	$\mu = 147.16$ $\sigma = 12.46$

The above table should be compared with the expected ℓ_1-norm and standard deviation of vectors sampled according to the CSIDH distribution, i.e. uniform random in $[-B, B]^n$. For $B = 5$ and $n = 74$, we obtain $\mu = n2(5+4+3+2+1)/11 = 201.81$ and $\sigma = 13.76$, but note $(2B+1)^{74} < N/2.2$ so less than half of the class group is covered by CSIDH.

To lower the ℓ_1-norm further, we can employ an algorithm due to Doulgerakis, Laarhoven and de Weger [14] (originally described in [27]). The idea of this algorithm is pretty simple: given a list $\mathcal{S}$ of short vectors in the lattice L, it tries to construct a vector that is closer to $\mathbf{e}$ than the current vector $\mathbf{z}$ by considering $\mathbf{z} \pm \mathbf{s}$ for all $\mathbf{s} \in \mathcal{S}$. This procedure is then repeated on small random shifts of the target vector. The resulting DLW algorithm is described in Algorithm 1.

Algorithm 1 DLW algorithm - randomized slicer for solving CVP

```
Input: A list $\mathcal{S} \subset L$ of short vectors, target vector $\mathbf{e} \in \mathbb{Z}^n$, number of iterations $M$
Output: Approximate closest lattice vector $\mathbf{z}$ to $\mathbf{e}$
 1: $\mathbf{z} \leftarrow \mathbf{0}$
 2: for $i = 0, \dots, M-1$ do
 3:     Randomize $\mathbf{e}$ with random small lattice vector to obtain $\mathbf{e}'$
 4:     for $\mathbf{s} \in \mathcal{S}$ do
 5:         if $\|\mathbf{e}' - \mathbf{s}\|_1 < \|\mathbf{e}'\|_1$ then
 6:             $\mathbf{e}' \leftarrow \mathbf{e}' - \mathbf{s}$ and restart for loop in line (4)
 7:         end if
 8:     end for
 9:     if $\|\mathbf{e}'\|_1 < \|\mathbf{e} - \mathbf{z}\|_1$ then
10:         $\mathbf{z} \leftarrow \mathbf{e} - \mathbf{e}'$
11:     end if
12: end for
13: return $\mathbf{z}$
```

We ran Algorithm 1 for varying sizes of lists of short vectors and varying number of iterations; the results can be found in Table 2.

Our experiments indicate that (on our setup) the fastest approach is to use the Babai nearest plane method with 2 iterations of the DLW algorithm, with a list of 10000 short vectors. In this case, the reduction takes $7.2 \cdot 10^6$ cycles on average, and evaluating the CSIDH action takes on average $128.1 \cdot 10^6$ cycles. In comparison, standard CSIDH-512 uses vectors sampled uniformly from $[-5, 5]^{74}$ (which does not sample uniformly from $\mathrm{Cl}(\mathcal{O})$) and takes on average $117.7 \cdot 10^6$ cycles. Hence, the additional cost of sampling uniformly is only 15%.

Table 2. ℓ_1-norm, ℓ_2-norm and evalutation time (reduction + action) of the DLW algorithm combined with Babai's nearest plane method on an HKZ basis

List size	Iterations	ℓ_1-norm	ℓ_2-norm	time of reduction + action
1000	1	223.54 ± 13.29	34.07 ± 2.45	140.17 ± 10.32
1000	3	221.38 ± 11.82	33.79 ± 2.26	138.02 ± 10.24
1000	10	216.84 ± 10.14	33.21 ± 2.03	137.66 ± 9.82
3000	1	219.02 ± 12.02	33.65 ± 2.34	138.09 ± 10.25
3000	3	214.96 ± 10.33	33.03 ± 2.09	136.78 ± 9.46
3000	10	208.75 ± 8.55	32.12 ± 1.81	136.95 ± 8.73
10000	1	213.96 ± 10.92	33.09 ± 2.30	135.55 ± 9.53
10000	3	207.97 ± 9.10	32.08 ± 1.93	135.41 ± 8.82
10000	10	201.26 ± 7.47	31.05 ± 1.66	144.26 ± 7.94

5 The signature scheme

In this section we propose CSI-FiSh, an efficient isogeny based signature scheme. The basis of CSI-FiSh was already sketched by Stolbunov in his thesis [35, 2.B]. He applies the Fiat-Shamir transform [16] to an isogeny based identification scheme by Couveignes [9] and independently by Stolbunov [34].

$$\begin{tikzcd}
 & E & \\
E_0 \arrow[ur, "\mathfrak{b}\star"] \arrow[rr, "\mathfrak{a}\star"] & & E_1 \arrow[ul, dashed, "\mathfrak{r}\star"']
\end{tikzcd}$$

Figure 1. The basic identification scheme for challenge $c = 1$.

5.1 The basic identification scheme

The identification scheme is illustrated in Figure 1 and works as follows: the public key of the prover consists of $E_1 = \mathfrak{a} \star E_0$ with $\mathfrak{a}$ a random element in $\mathrm{Cl}(\mathcal{O})$ and E_0 the base curve specified by the system parameters. Assuming that $\mathrm{Cl}(\mathcal{O})$ is cyclic with generator $\mathfrak{g}$, we can write $\mathfrak{a} = \mathfrak{g}^a$ with a random in $\mathbb{Z}_N$ and $N = \#\mathrm{Cl}(\mathcal{O})$. The prover samples a random element $\mathfrak{b} = \mathfrak{g}^b$ with $b \in_R \mathbb{Z}_N$ and commits to the (isomorphism class of the) curve $E = \mathfrak{g}^b \star E_0 = [b]E_0$. The verifier then chooses a random bit $c \in \{0,1\}$ and sends this to the prover. If $c = 0$, the prover responds with $r = b$, and the verifier checks that $E = [r]E_0$, if $c = 1$, the prover responds with $r = b - a \bmod N$ and the verifier checks that $E = [r]E_1$. Note that reducing modulo N is required to avoid any leakage on a and that the check can be written as $E = [r]E_c$. A detailed description of the protocol is displayed in Figure 2.

Prover		**Verifier**
$b \leftarrow_R \mathbb{Z}_N$ $E \leftarrow [b]E_0$		
	$\xrightarrow{\mathsf{E}}$	
		$c \overset{\$}{\leftarrow} \{0,1\}$
	$\xleftarrow{c}$	
$r \leftarrow b - c \cdot a \bmod N$		
	$\xrightarrow{r}$	
		return $E \overset{?}{=} [r]E_c$

Figure 2. The identification scheme of Couveignes and Stolbunov.

Theorem 4. *The Couveignes-Stolbunov protocol (Figure 2) is a complete and secure Sigma protocol proving knowledge of a solution of a GAIP instance. That is, it enjoys completeness, special soundness and special Honest-Verifier Zero Knowledge.*

Proof. **Completeness.** Suppose the protocol is followed honestly, and suppose $E_1 = [a]E_0$. In the case $c = 0$ the verifier checks if $E = [b]E_0$, which is true by construction of E. In the case $c = 1$ the verifier checks if $E = [b-a]E_1$ which holds because

$$[b-a]E_1 = [b-a][a]E_0 = [b]E_0 = E\,.$$

Special Soundness. Suppose $(E, 0, r_0)$ and $(E, 1, r_1)$ are two transcripts that are accepted by the verifier. Then we have

$$E = [r_0]E_0 = [r_1]E_1\,,$$

from which it follows that $[r_0 - r_1]E_0 = E_1$. Hence, it is trivial to extract $r_0 - r_1$, which is a solution to the GAIP problem.

Special Honest-Verifier Zero Knowledge. Consider the simulator that, given a bit c picks a random $r \in \mathbb{Z}_N$, computes $E = [r]E_c$ and outputs the transcript (E, c, r). Then it is clear that the transcripts generated by the simulator are indistinguishable from transcripts of honest executions of the protocol with challenge equal to c: both the real transcripts and the simulated transcripts have uniformly random distributed values of r, and $E = [r]E_c$. □

5.2 Optimizing the Sigma protocol

Hashing. To reduce the communication cost (and hence the signature size after applying the Fiat-Shamir transform) it suffices for the Prover to send $\mathcal{H}(E)$ rather than E, for some collision resistant hash function $\mathcal{H}$. The verifier then computes $\mathcal{H}([r]E_c)$ and checks that it is equal to the hash value sent by the prover. If we are doing t rounds of the protocol in parallel to amplify soundness, it suffices to send a single hash of the concatenation of all the $E^{(i)}$ for i from 1 to t. Clearly the completeness and the Honest-Verifier Zero Knowledge properties of the scheme are not affected by this change. For special soundness, the collision resistance of $\mathcal{H}$ implies that if

$$\mathcal{H}([r_1^{(1)}]E_{c_1^{(1)}}||\cdots||[r_1^{(t)}]E_{c_1^{(t)}}) = \mathcal{H}([r_2^{(1)}]E_{c_2^{(1)}}||\cdots||[r_2^{(t)}]E_{c_2^{(t)}})$$

then $[r_1^{(i)}]E_{c_1^{(i)}} = [r_2^{(i)}]E_{c_2^{(i)}}$ for all i from 1 to t. Hence, if we model $\mathcal{H}$ as a random oracle it is sufficient for $\mathcal{H}$ to have output length 2λ, with λ the security level.

Larger challenge spaces. A well-known approach [11] to lower the soundness error is to increase the challenge space. To do this we move from the GAIP problem to the MT-GAIP problem. We now have $S - 1$ public keys instead of one, i.e. the public key now consists of the S-tuple $(E_0, E_1 = [a_1]E_0, \ldots, E_{S-1} = [a_{S-1}]E_0)$ (note that E_0 can be left out, it is just there to illustrate the notation) and the prover proves to the verifier that he knows an $s \in \mathbb{Z}_N$ such that $[s]E_i = E_j$ for some pair of curves in the public key (with $i \neq j$). The prover still chooses a random exponent $b \in_R \mathbb{Z}_N$ and computes $E^{(i)} = [b]E_0$. The verifier now sends a challenge $c \in [0, S[$, and the response consists of $r = b - a_c \bmod N$. The verifier then recomputes $[r]E_c$ and verifies that this is equal to $E^{(i)}$. Theorem 4 generalizes to the new identification scheme. In particular, since the challenge space now contains S elements the soundness error drops to $1/S$.

Theorem 5. *The adapted identification scheme is a complete and secure Sigma protocol proving knowledge of a solution of an MT-GAIP instance.*

Proof. The proof is completely analogous to the proof of Theorem 4. □

Doubling the challenge space with twists. To increase the size of the challenge space even further, we exploit the fact that given a curve $E = [a]E_0$, its quadratic twist E^t (which can be computed very efficiently) is $\mathbb{F}_p$-isomorphic to $[-a]E_0$ [6]. Therefore, we can almost double the set of public key curves going from $E_0, E_1, \ldots, E_{S-1}$ to $E_{-S+1}, \ldots, E_0, \ldots, E_{S-1}$, where $E_{-i} = E_i^t$, without any increase in communication cost. Hence, the soundness error drops to $\frac{1}{2S-1}$. Theorem 5 still applies, but instead of a reduction from a random MT-GAIP instance, we now have a reduction from a random MT-GAIP instance subject to $E_{-i} = E_i^t$ (we call this twisted MT-GAIP). However, there is a simple reduction from this problem to GAIP, which shows this optimization does not affect security.

Theorem 6. *Given an adversary $\mathcal{A}$ that solves a random instance of twisted MT-GAIP in time T and with probability ϵ, there exists an adversary $\mathcal{B}^{\mathcal{A}}$ that solves a random instance of MT-GAIP in time $T + O(S)$ with probability at least $\epsilon/2$.*

Proof. We describe the adversary $\mathcal{B}^{\mathcal{A}}$. Suppose $\mathcal{B}$ is given a random MT-GAIP instance $E_1, \ldots, E_k$, then he chooses k random bits $b_1, \ldots, b_k$ and defines curves

$$\tilde{E_i} = \begin{cases} E_i \text{ if } b_i = 0 \\ E_i^t \text{ if } b_i = 1 \end{cases},$$

then he sets $\tilde{E_0} = E_0$ and $\tilde{E}_{-i} = \tilde{E_i}^t$ for all i in $\{1, \ldots, k\}$. This is a random twisted MT-GAIP instance that $\mathcal{B}$ then sends to $\mathcal{A}$. With probability ϵ, $\mathcal{A}$ responds with (a, i, j) such that $i \neq j$ and $\tilde{E_i} = [a]\tilde{E_j}$. Now we consider 2 cases:

- $i = -j$. In this case we have $\tilde{E_i} = [a]\tilde{E_i}^t$, which implies $\tilde{E_i} = [a/2]E_0$, so $\mathcal{B}$ outputs $((-1)^{b_{|i|}}a/2, |i|, 0)$, which is a valid solution to his MT-GAIP instance ($|\mathrm{Cl}(\mathcal{O})|$ is known to be odd, so the inverse of 2 always exists).

- $|i| \neq |j|$. In this case we have $\text{sign}(i)(-1)^{b_{|i|}} = \text{sign}(j)(-1)^{b_{|j|}}$ with probability $\frac{1}{2}$. In this case we have an equation of the form $E_i = [\pm a]E_j$ or $E_i^t = [\pm a]E_j^t$. Therefore B can output a valid solution to his MT-GAIP problem $(\pm a, |i|, |j|)$.

Shorter public keys. The previous section explains how one can improve the communication cost and the proving and verification time by considering multiple public key curves $E_i = [a_i]E_0$ for $i \in \{1, \dots, S-1\}$. The drawback of this approach is that the public key now consists of $S-1$ curves, so its size blows up as S increases. Note that at most t of these public key curves are used during each verification (where t is the number of parallel executions of the protocol to amplify soundness). Therefore, instead of including all the curves $E_1, \dots, E_{S-1}$ in the public key, the public key can just be a commitment to those curves. The improvement in total communication cost comes from the fact that the response of the prover now only has to include the opening of at most t curves $E_{c_1}, \dots, E_{c_t}$. If the commitment scheme is binding, then a cheating prover cannot open the commitment to an incorrect curve, so the security of the scheme is preserved. We use a Merkle tree construction to implement the binding commitments, because this allows for the efficient opening of a subset of the curves.

In particular, suppose for simplicity that $S - 1 = 2^d$ and let

$$h_{d,i} = \mathcal{H}(E_i || 2^d + i || \mathsf{MerkleKey}) \,,$$

where $\mathsf{MerkleKey} \in \{0,1\}^\lambda$ is a key which is chosen uniformly at random during key generation and included in both the secret and public keys. Then we define each internal node of the Merkle tree as the hash of its children, concatenated with its position in the tree and the $\mathsf{MerkleKey}$:

$$h_{k,i} = \mathcal{H}(h_{k+1,2i-1} || h_{k+1,2i} || 2^k + i || \mathsf{MerkleKey}) \,.$$

It is an easy exercise to show that if we model $\mathcal{H}$ as a random oracle, the root of the Merkle tree is a binding commitment: An adversary making q queries to the random oracle has at most probability $\frac{q+1}{2^\lambda}$ of breaking the binding property. Note that the $\mathsf{MerkleKey}$ is not strictly required to prove soundness, but it prevents an adversary from attacking multiple public keys at the same time. A similar approach of reducing the public key size was proposed by [11]. They use the more complicated and slightly less efficient construction of [19], which is designed to be provably secure in the standard model. Since the Fiat-Shamir transform relies on the (Q)ROM anyway, there is no reason to use this approach.

5.3 Signatures

The above identification schemes can be turned into (non-interactive) signature schemes using the Fiat-Shamir transform [16], where the challenges $c_i \in \{-S+1, \dots, S-1\}$ are simply obtained by hashing the ephemeral keys $E^{(i)}$ for $i = 1, \dots, t$ together with the message m, i.e. $(c_1, \dots, c_t) = \mathcal{H}(E^{(1)}|| \cdots ||E^{(t)}||m)$. The signature then consists of $(r_1, \dots, r_t, c_1, \dots, c_t)$, and the verifier recomputes the $E^{(i)} = [r_i]E_{c_i}$ and checks that indeed $(c_1, \dots, c_t) = \mathcal{H}(E^{(1)}|| \cdots ||E^{(t)}||m)$. Figure 3 details the "simple" variant and corresponds to the identification scheme using multiple public keys. The "Merkle" variant reduces the size of the public key by using a Merkle tree as described above.

To achieve security level λ, we require $t = \lambda / \log_2 S$ and the resulting signature size is $t(\lceil \log_2 N \rceil + \lceil \log_2 S \rceil)$ bits for the simple variant. The "Merkle" variant needs to include the openings of merkle paths in the signature, the total size of these openings depends on the leaves that are opened. For example, in the extremely unlikely case that all the t challenges are identical only one Merkle path needs to be opened. Both signing and verification require t CSIDH actions (including the time to construct a small representant of the ideal).

The results on Fiat-Shamir in the QROM of Don et al. [13] readily apply to our setting:

Theorem 7. *Assume the hash functions used are modeled as quantum random oracles, then CSI-FiSh is sEUF-CMA secure.*

Proof. The basic sigma protocol (without hashing) has special soundness and unique responses (for each i there exists only one value of $r_i \in \mathbb{Z}_N$ such that $[r_i]E_{c_i} = E^{(i)}$). Hence, Theorem 25 of [13] implies that the scheme also has the Quantum Proof of Knowledge property. The protocol also has more than λ bits of min entropy and perfect HVZK, so Theorem 22 of [13] implies that the Fiat-Shamir scheme is sEUF-CMA secure in the QROM.

For the variant with hashing, it is known that Quantum random oracles are collapsing, so it is immediate that the sigma protocol has quantum computationally unique responses. Hence, the claim again follows from Theorems 25 and 22 of [13].

Algorithm 2 KeyGen

```
Input: $E_0$, class number $N = \#\mathrm{Cl}(\mathcal{O})$
Output: $\mathbf{sk}, \mathbf{pk}$
 1: for $i \in \{1, \dots, S-1\}$ do
 2:     $a_i \leftarrow_R \mathbb{Z}_N$
 3:     $E_i = [a_i]E_0$
 4: end for
 5: $\mathbf{pk} = [E_i : i \in \{1, \dots, S-1\}]$
 6: return $(\mathbf{sk} = \mathbf{a}, \mathbf{pk})$
```

Algorithm 3 Sign

```
Input: $\mathrm{msg}, \mathbf{sk} = \mathbf{a}$
Output: $\sigma = (r_1, \dots, r_t, c_1, \dots, c_t)$
 1: $a_0 \leftarrow 0$
 2: for $i = 1, \dots, t$ do
 3:     $b_i \leftarrow_R \mathbb{Z}_N$, $E^{(i)} = [b_i]E_0$
 4: end for
 5: $(c_1, \dots, c_t) = \mathcal{H}(E^{(1)}||\cdots||E^{(t)}||m)$
 6: for $i = 1, \dots, t$ do
 7:     $r_i = b_i - \mathrm{sign}(c_i)a_{|c_i|} \bmod N$
 8: end for
 9: return $\sigma = (r_1, \dots, r_t, c_1, \dots, c_t)$
```

Algorithm 4 Verify

```
Input: $\mathrm{msg}, E_0, \mathbf{pk} = [E_i : i \in \{1, \dots, S-1\}], \sigma$
Output: Valid / invalid
 1: Parse $\sigma$ as $(r_1, \dots, r_t, c_1, \dots, c_t)$
 2: Define $E_{-i} = E_i^t$ for all $i \in \{1, \dots, S-1\}$.
 3: for $i = 1, \dots, t$ do
 4:     $E^{(i)} = [r_i]E_{c_i}$
 5: end for
 6: $(c'_1, \dots, c'_t) = \mathcal{H}(E^{(1)}||\cdots||E^{(t)}||m)$
 7: if $(c_1, \dots, c_t) == (c'_1, \dots, c'_t)$ then
 8:     return Valid
 9: else
10:     return Invalid
11: end if
```

6 Implementation results

6.1 Parameter Choices

Slow Hash functions Because the QROM security proof is very non-tight it would not be practical to choose parameters in such a way that security is guaranteed by the proof. Instead, as is customary, we assume that the probablity of a successful attack is at most $Q \times E$, where Q is the number of hash function evaluations that an attacker makes, and E is the soundness error of the zero knowledge proof. So usually one would choose the parameters S and t such that $S^{-t} \leq 2^{-\lambda}$. In our implementation we choose a hash function that is a factor 2^k slower than a standard hash function (e.g. SHA-3), therefore it suffices to take our parameters such that $S^{-t} \leq 2^{-\lambda+k}$. We pick k in such a way that the time spent evaluating the slow hash function is small compared to the total signing and verification time. Since we can take smaller parameters this optimization slightly reduces both the signature size and the signing and verification time.

Proposed parameter sets We have implemented several parameter sets for both the "simple" variant and the "Merkle" variant. For the simple variant the secret key is always small and the variable S controls a trade-off between on the one hand small public keys and fast key generation (when S is small), and on the other hand small signatures and fast signing and verification (when S is large). When we use the "Merkle" variant the public key is always small, but the secret key size increases with increasing value of S, because we store the entire Merkle tree to avoid having to recompute the public keys during signing.

6.2 Implementation details and Benchmarking results

Our proof-of-concept implementation is available on GitHub [2]. To evaluate the CSIDH action, we use the `20180826` version of the proof-of-concept implementation by Castryck et al. [6]. Our implementation depends on the eXtended Keccak Code Package for the implementation of SHAKE256, which we have used as hash function, commitment scheme and to expand randomness. The implementation of the Babai nearest plane step depends on the GMP library for its high precision arithmetic. Since we rely on the implementation of Castryck et al. [6], the implementation is not constant-time. Implementing an optimized

Table 3. Parameter choices and benchmark results for the "simple" variant of CSI-FiSh .

S	t	k	$\|\mathbf{sk}\|$	$\|\mathbf{pk}\|$	$\|\mathbf{sig}\|$	KeyGen	Sign	Verify
2^1	56	16	16 B	128 B	1880 B	100 ms	2.92 s	2.92 s
2^2	38	14	16 B	256 B	1286 B	200 ms	1.98 s	1.97 s
2^3	28	16	16 B	512 B	956 B	400 ms	1.48 s	1.48 s
2^4	23	13	16 B	1 KB	791 B	810 ms	1.20 s	1.19 s
2^6	16	16	16 B	4 KB	560 B	3.3 s	862 ms	859 ms
2^8	13	11	16 B	16 KB	461 B	13 s	671 ms	670 ms
2^{10}	11	7	16 B	64 KB	395 B	52 s	569 ms	567 ms
2^{12}	9	11	16 B	256 KB	329 B	3.5 m	471 ms	469 ms
2^{15}	7	16	16 B	2 MB	263 B	28 m	395 ms	393 ms

Table 4. Parameter choices and benchmark results for the "Merkle" variant of CSI-FiSh .

S	t	k	$\|\mathbf{sk}\|$	$\|\mathbf{pk}\|$	$\|\mathbf{sig}\|$	KeyGen	Sign	Verify
2^8	13	11	8 KB	32 B	1995 B	13 s	671 ms	371 ms
2^{10}	11	7	32 KB	32 B	2086 B	52 s	567 ms	567 ms
2^{12}	9	11	128 KB	32 B	2022 B	3.5 m	467 ms	467 ms
2^{15}	7	16	1 MB	32 B	1953 B	28 m	399 ms	402 ms
2^{18}	6	14	8 MB	32 B	1990 B	3.8 h	335 ms	326 ms

constant-time implementation of CSI-FiSh is outside the scope of this paper and is left for future work.

All our benchmarking experiments are performed on a Dell OptiPlex 3050 machine with Intel Core i5-7500T CPU @ 2.70GHz. The benchmarking results are displayed in Tables 3 and 4.

Remark 8. Like most discrete logarithm based signature schemes, it is possible to precompute the ephemeral keys in CSI-FiSh, i.e. all CSIDH actions can be computed offline, and the online phase then only consists of t modular subtractions, which are extremely fast.

7 Conclusions and open problems

We computed the class group of the imaginary quadratic field that is at the heart of the CSIDH-512 cryptosystem, and exploited the knowledge of the relation lattice to instantiate the first efficient isogeny based signature scheme called CSI-FiSh. The scheme is flexible in that it allows trade-offs between signature sizes, key sizes and the time to sign/verify. One parameter set of CSI-FiSh gives the smallest combined size of public key and signature, compared to any other existing post-quantum secure signature scheme at the 128-bit security level.

Should the CSIDH-512 parameters turn out to be insufficiently secure, then the class group computation in this paper can be repeated for a larger prime. Even though the computation for the CSIDH-512 parameters already broke previous records, the effort of 52 core years is relatively small compared to other record computations such as for factoring and DLP, which often take thousands of core years. Our computation took less than a month with the resources available to us. Hence, there is still quite some room to compute class groups for increased parameters. Moreover, the class group can be computed in quantum polynomial time. Hence, it seems likely that quantum computers that can compute large class groups will be available well before there are quantum computers that can break CSIDH-512.

The main open problem, given that the class group is cyclic of order N, is to devise an identification scheme where the challenge is taken from $\mathbb{Z}_N$, instead of binary or from the small set $]-S,S[$. Note that the prover can simply mimick the discrete logarithm based constructions since he can now work in the *ring* $\mathbb{Z}_N$, and thus can create the typical response expressing a combination of the ephemeral key, secret key and challenge. The major problem however is how the verifier can verify this combination to be correct, since the group action still only allows to add a known constant in $\mathbb{Z}_N$. The impact of such an identification scheme would be major: the signature size could possibly be as small as 64 bytes, the public key also 64 bytes and signing would require only one CSIDH action taking around 40ms.

Acknowledgements

We would like to thank the department of Electrical Engineering at KU Leuven for providing the necessary computing power through the HTCondor framework. Many thanks also to Léo Ducas for computing the HKZ basis of the relation lattice and discussions on CVP solvers.

References

[1] László Babai. On Lovász' lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13, 1986.

[2] Ward Beullens. CSI-FiSh: github repository available at `https://github.com/KULeuven-COSIC/CSI-FiSh`, 2019.

[3] Ward Beullens and Bart Preneel. Field lifting for smaller UOV public keys. In *International Conference on Cryptology in India*, pages 227–246. Springer, 2017.

[4] Jean-François Biasse. Improvements in the computation of ideal class groups of imaginary quadratic number fields. *Adv. in Math. of Comm.*, 4(2):141–154, 2010.

[5] Xavier Bonnetain and André Schrottenloher. Submerging csidh. Cryptology ePrint Archive, Report 2018/537, 2018. `https://eprint.iacr.org/2018/537`.

[6] Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes. CSIDH: An Efficient Post-Quantum Commutative Group Action. In *ASIACRYPT*, volume 11274 of *Lecture Notes in Computer Science*, pages 395–427. Springer, 2018. `https://ia.cr/2018/383`.

[7] Andrew M. Childs, David Jao, and Vladimir Soukharev. Constructing elliptic curve isogenies in quantum subexponential time. *J. Mathematical Cryptology*, 8(1):1–29, 2014.

[8] Don Coppersmith. Solving homogeneous linear equations over GF(2) via block Wiedemann algorithm. *Mathematics of Computation*, 62:333–350, 1994.

[9] Jean Marc Couveignes. Hard Homogeneous Spaces., 1997. IACR Cryptology ePrint Archive 2006/291, `https://ia.cr/2006/291`.

[10] Luca De Feo. Mathematics of isogeny based cryptography, 2017. `https://defeo.lu/ema2017/poly.pdf`.

[11] Luca De Feo and Steven D Galbraith. Seasign: Compact isogeny signatures from class group actions. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 759–789. Springer, 2019.

[12] Thomas Decru, Lorenz Panny, and Frederik Vercauteren. Faster seasign signatures through improved rejection sampling. In *International Conference on Post-Quantum Cryptography*, pages 271–285. Springer, 2019.

[13] Jelle Don, Serge Fehr, Christian Majenz, and Christian Schaffner. Security of the Fiat-Shamir transformation in the quantum random-oracle model. *arXiv preprint arXiv:1902.07556*, 2019.

[14] Emmanouil Doulgerakis, Thijs Laarhoven, and Benne de Weger. Finding closest lattice vectors using approximate Voronoi cells. *PQCRYPTO. Springer*, 2019.

[15] Luca De Feo, David Jao, and Jérôme Plût. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. *J. Mathematical Cryptology*, 8(3):209–247, 2014.

[16] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1986.

[17] Steven D. Galbraith, Christophe Petit, and Javier Silva. Identification Protocols and Signature Schemes Based on Supersingular Isogeny Problems. In *Advances in Cryptology - ASIACRYPT 2017*, volume 10624 of *Lecture Notes in Computer Science*, pages 3–33. Springer, 2017.

[18] James L. Hafner and Kevin S. McCurley. A rigorous subexponential algorithm for computation of class groups. *Journal of the American Mathematical Society*, 2:837–850, 1989.

[19] Andreas Hülsing, Joost Rijneveld, and Fang Song. Mitigating multi-target attacks in hash-based signatures. In *Public-Key Cryptography–PKC 2016*, pages 387–416. Springer, 2016.

[20] Michael J. Jacobson. Applying sieving to the computation of quadratic class groups. *Math. Comp*, 68:859–867, 1999.

[21] David Jao, Reza Azarderakhsh, Matthew Campagna, Craig Costello, Luca De Feo, Basil Hess, Amir Jalali, Brian Koziel, Brian LaMacchia, Patrick Longa, Michael Naehrig, Joost Renes, Vladimir Soukharev, and David Urbanik. SIKE, 2016. Submission to [29]. `http://sike.org`.

[22] David Jao and Luca De Feo. Towards Quantum-Resistant Cryptosystems from Supersingular Elliptic Curve Isogenies. In *PQCrypto*, volume 7071 of *Lecture Notes in Computer Science*, pages 19–34. Springer, 2011. `https://ia.cr/2011/506`.

[23] Thorsten Kleinjung. Quadratic sieving. *Mathematics of Computation*, 85(300):1861–1873, 2016.

[24] Aleksandr Korkine and G Zolotareff. Sur les formes quadratiques. *Mathematische Annalen*, 6(3):366–389, 1873.

[25] Greg Kuperberg. A Subexponential-Time Quantum Algorithm for the Dihedral Hidden Subgroup Problem. *SIAM J. Comput.*, 35(1):170–188, 2005.

[26] Greg Kuperberg. Another Subexponential-time Quantum Algorithm for the Dihedral Hidden Subgroup Problem. In *TQC*, volume 22 of *LIPIcs*, pages 20–34. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2013.

[27] Thijs Laarhoven. Sieving for closest lattice vectors (with preprocessing). In *Selected Areas in Cryptography - SAC 2016*, volume 10532 of *Lecture Notes in Computer Science*, pages 523–542. Springer, 2017.

[28] Arjen Klaas Lenstra, Hendrik Willem Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, 1982.

[29] National Institute of Standards and Technology. Post-Quantum Cryptography Standardization, December 2016. `https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Post-Quantum-Cryptography-Standardization`.

[30] Chris Peikert. He gives c-sieves on the csidh. Cryptology ePrint Archive, Report 2019/725, 2019. `https://eprint.iacr.org/2019/725`.

[31] Alexander Rostovtsev and Anton Stolbunov. Public-Key Cryptosystem Based on Isogenies, 2006. IACR Cryptology ePrint Archive 2006/145. `https://ia.cr/2006/145`.

[32] Claus-Peter Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. *Theoretical computer science*, 53(2-3):201–224, 1987.

[33] Joseph H Silverman. *The Arithmetic of Elliptic Curves.* Graduate Texts in Mathematics. Springer, Dordrecht, 2009.

[34] Anton Stolbunov. Constructing public-key cryptographic schemes based on class group action on a set of isogenous elliptic curves. *Adv. in Math. of Comm.*, 4(2):215–235, 2010.

[35] Anton Stolbunov. Cryptographic schemes based on isogenies, Doctoral thesis, NTNU, 2012.

[36] Jacques Vélu. Isogénies entre courbes elliptiques. *CR Acad. Sci. Paris, Séries A*, 273:305–347, 1971.

[37] Douglas H. Wiedemann. Solving sparse linear equations over finite fields. *IEEE Trans. Information Theory*, 32(1):54–62, 1986.

[38] Youngho Yoo, Reza Azarderakhsh, Amir Jalali, David Jao, and Vladimir Soukharev. A Post-quantum Digital Signature Scheme Based on Supersingular Isogenies. In *Financial Cryptography and Data Security - FC 2017*, volume 10322 of *Lecture Notes in Computer Science*, pages 163–181. Springer, 2017.

Chapter 13

LUOV

Do not pity the dead, Harry. Pity those who live without LUOV.

– Albus Dumbledore

Publication Data

Beullens W., Preneel B. Field Lifting for Smaller UOV Public Keys. In: Progress in Cryptology – INDOCRYPT 2017. Lecture Notes in Computer Science, vol 10698, pages 227-246. Springer, 2017.

My Contribution

Main author. I designed the signature scheme, did most of the writing work, wrote the implementation and did the benchmarking.

Field lifting for smaller UOV public keys

Ward Beullens and Bart Preneel

imec-COSIC KU Leuven,
Kasteelpark Arenberg 10 - bus 2452, 3001 Heverlee, België
ward.beullens@esat.kuleuven.be,
bart.preneel@esat.kuleuven.be

Abstract. Most Multivariate Quadratic (MQ) signature schemes have a very large public key, which makes them unsuitable for many applications, despite attractive features such as speed and small signature sizes. In this paper we introduce a modification of the Unbalanced Oil and Vinegar (UOV) signature scheme that has public keys which are an order of magnitude smaller than other MQ signature schemes. The main idea is to choose UOV keys over the smallest field $\mathbb{F}_2$ in order to achieve small keys, but to lift the keys to a large extension field, where solving the MQ problem is harder. The resulting Lifted UOV signature scheme is very competitive with other post-quantum signature schemes in terms of key sizes, signature sizes and speed.

Keywords: Post-Quantum Cryptography, Multivariate Cryptography, Signature Schemes, Unbalanced Oil and Vinegar, Key Size Reduction

1 Introduction

When large scale quantum computers are built, they will be able to break nearly all public key cryptography that is being used today, including RSA [25], DSA [17] and ECC. This is because these schemes rely on the hardness of number theoretic problems such as integer factorization and finding discrete logarithms, which can be solved efficiently by Shor's Algorithm [26]. Even if it would take 10 or 20 years to build large scale quantum computers, upgrading our current systems may be very slow and some stored data requires long term protection (in particular for confidentiality). To avert a potential catastrophe, post-quantum cryptography should be designed, implemented and deployed well before large scale quantum computers are built.

During recent years, the research on post-quantum cryptography has been accelerating. One of the goals of the EU-funded PQCRYPTO project is to develop and standardize post-quantum algorithms [1]. Recently NIST, the US National Institute for Standards and Technology, has started the process of selecting post-quantum algorithms for standardization [19]. According to both PQCRYPTO and NIST, multivariate cryptography is one of the major candidates for providing post-quantum security. Multivariate cryptography is based on the hardness of some problems related to multivariate polynomials over finite fields, such as solving multivariate polynomial equations. In general, multivariate cryptography is very fast and requires only moderate computational resources, which makes it attractive for applications in low-cost devices. However, a disadvantage of multivariate cryptography is its large public keys, which can be prohibitive for many applications. Some work in mitigating this problem in the case of the UOV and Rainbow signature schemes has been published by Petzoldt [22], who managed to reduce the key size by a factor of 8 in the case of UOV and a factor of 3 in the case of the Rainbow signature scheme. His proposal makes a small modification to the key generation algorithm and exploits the fact that a large part of the public key can be freely chosen by the user. One can then choose to generate this part using a Pseudo-Random Number Generator (PRNG), and to only store the seed for the PRNG. In this paper we introduce a new idea to reduce the size of the public keys of UOV dramatically, by lifting the public and central maps to an extension field. The new idea is compatible with the ideas of Petzoldt and together they provide public keys that are up to 10 times smaller than if we were to use only Petzoldt's modification of UOV.

Before introducing the Lifted UOV signature scheme in Sect. 5, we present an overview of the MQ problem in Sect. 2 and the UOV signature and how it was improved by Pezoldt in Sects. 3 and 4. We finish with a brief description of our software implementation in Sect. 6 and conclude in Sect. 7.

2 The MQ problem

The security of an MQ signature scheme relies on the hardness of the MQ-problem. We give a brief discussion of the problem here.

MQ Problem. Given a quadratic polynomial map $\mathcal{P} : \mathbb{F}_q^n \to \mathbb{F}_q^m$ over a finite field $\mathbb{F}_q$, find $\mathbf{x} \in \mathbb{F}_q^n$ that satisfies $\mathcal{P}(\mathbf{x}) = \mathbf{0}$.

It is known that the MQ problem is NP-hard [18]. Therefore it is unlikely that there are (quantum) algorithms that solve the hardest instances of the MQ problem in polynomial time. The problem is also believed to be hard on average in the case $n \approx m$. Only exponential time algorithms are known to solve random instances of the problem for these parameters.

Systems with $n = m$ are called determined systems; these are the most difficult systems to solve. When $n < m$ a system is called overdetermined, and when $n > m$ the system is called underdetermined. Thomae et al. showed that finding a solution for an underdetermined system with $n = \alpha m$ can be reduced to finding a solution of a determined system with only $m + 1 - \lfloor \alpha \rfloor$ equations [27]. This means that as a system becomes more underdetermined it becomes easier to solve. This fact will become important in the security analysis of UOV.

2.1 Classical algorithms

The best known classical algorithms to solve the MQ-problem for generic determined systems over finite fields use the hybrid approach [5, 6]. This approach combines exhaustive search with Gröbner basis computations. In this approach k variables are fixed to random values and the remaining $n - k$ variables are found with a Gröbner basis algorithm such as F_4, F_5 or XL. If no assignment to the remaining $n - k$ variables exists that solves the system, the procedure starts again with a different guess for the first k variables. We require on average roughly q^k Gröbner basis computations until a solution is found. As a result, the optimal value of k decreases as q increases. The complexity of computing a Gröbner basis for a system of polynomials depends critically on the degree of regularity (d_{reg}) of that system. Though it is of little importance to the rest of the paper, we refer to Bardet [2] for a precise definition of the degree of regularity. The complexity of the F_5 algorithm is given by

$$C_{F_5}(n, d_{reg}) = O\left(\binom{n + d_{reg}}{d_{reg}}^{\omega}\right),$$

where $2 \leq \omega < 3$ is the constant in the complexity of matrix multiplication. Therefore the complexity of the hybrid approach is

$$C_{\mathrm{Hybrid}F_5(n,d_{reg},k)} = O\left(q^k \binom{n - k + d_{reg}(k)}{d_{reg}(k)}^{\omega}\right), \tag{1}$$

where $d_{reg}(k)$ stand for the degree of regularity of the system after fixing the values of k variables.

Determining the degree of regularity for a specific polynomial system is difficult, but for a certain class of systems, called semi-regular systems, it is known that the degree of regularity can be deduced from the number of equations m and the number of variables n [2, 8]. In particular, for quadratic semi-regular systems the degree of regularity is the degree of the first term in the power series of

$$S_{m,n}(x) = \frac{(1-x^2)^m}{(1-x)^n}$$

with a non-positive coefficient. This gives a practical method to calculate the degree of regularity of any semi-regular system. Empirically, polynomial systems that are randomly chosen have a very large probability of being semi-regular and it is conjectured that most systems are semi-regular systems. For the definition and the theory of semi-regular systems we refer to chapter 3 of the PhD thesis of Bardet [2].

2.2 Quantum algorithms

Currently, there are no specialized quantum algorithms that solve polynomial systems over finite fields. However, Grover's algorithm [13] can be used to speed up the brute force part of the hybrid approach. This approach gives a quadratic speedup for the brute force part of the attack, so the new complexity would be

$$C_{\text{Hybrid}F_5(n,d_{reg},k)} = O\left(q^{k/2}\binom{n-k+d_{reg}(k)}{d_{reg}(k)}^{\omega}\right), \tag{2}$$

where the difference with (1) is that we have the factor $q^{k/2}$ instead of q^k. However it should be noted that this approach requires sequentially running $q^{k/2}$ Gröbner basis computations on a quantum computer. This would be an incredible feat because even for moderately sized polynomial systems this would require gigabytes worth of qubits and days of computation without decoherence. Also, note that the gains of parallelizing Grover search grow only with the square root of the number of independent computers used, instead of a linear growth for the classical brute force search [28]. Nevertheless, in the security analysis of the signature scheme proposed in this paper we will be cautious and assume that these kinds of attacks on the MQ problem are possible and we will make our parameter choices accordingly. This has the additional benefit of providing a large safety margin against classical attacks.

Remark 1. Typically the optimal value of k, i.e. the number of variables that is guessed by brute force, is quite small (eg. 2,3 or 4), this does *not* mean that the hybrid approach is only a marginal improvement over a direct Gröbner basis computation. Guessing only a few variables can drastically reduce the degree of regularity of a system. For example, guessing only one variable in a determined semi-regular system of polynomials roughly reduces the degree of regularity by half! The idea of lifting a public key to an extension field is a countermeasure to the hybrid approach. By working in a large extension field (eg. $\mathbb{F}_{2^{64}}$) we ensure that guessing even a single variable is computationally too expensive.

3 The UOV signature scheme

The UOV or Unbalanced Oil and Vinegar digital signature scheme is a multivariate quadratic (MQ) signature scheme. It is a slightly modified version of the original Oil and Vinegar signature scheme that was proposed by Patarin in 1997 [20]. With the right parameter choices UOV has withstood all cryptanalysis since 1997 and it is one of the best studied and most promising MQ signature schemes.

3.1 Description of UOV

The UOV signature scheme uses a one-way function $\mathcal{P} : \mathbb{F}_q^n \to \mathbb{F}_q^m$, which is a multivariate quadratic polynomial map over some finite field $\mathbb{F}_q$. The trapdoor is a factorization $\mathcal{P} = \mathcal{F} \circ \mathcal{T}$, where $\mathcal{T} : \mathbb{F}_q^n \to \mathbb{F}_q^n$ is an invertible linear map, and $\mathcal{F} : \mathbb{F}_q^n \to \mathbb{F}_q^m$ is a quadratic map whose components $f_1, \cdots, f_m$ are of the form

$$f_k(\mathbf{x}) = \sum_{i=1}^{v} \sum_{j=i}^{n} \alpha_{i,j,k} x_i x_j + \sum_{i=1}^{n} \beta_{i,k} x_i + \gamma_k \,,$$

where $v = n - m$. We say that the first v variables $x_1, \cdots, x_v$ are the vinegar variables, whereas the remaining m variables are the oil variables. The components of $\mathcal{F}$ are quadratic polynomials in the variables x_i such that there are no quadratic terms which contain two oil variables. One could say that the vinegar variables and the oil variables are not fully mixed, which is where their names come from. [1]

[1] However it is not a very good name because in reality oil mixes with oil and vinegar mixes with vinegar but no mixing happens between oil and vinegar, and this is not what happens in UOV polynomials. A better name would have been hen variables and rooster variables because hens can get along with hens and roosters, but two roosters start a fight when they appear in the same term. Moreover, this foreshad-

How does the trapdoor $\mathcal{P} = \mathcal{F} \circ \mathcal{T}$ help to invert the function $\mathcal{P}$? Given a target $\mathbf{x} \in \mathbb{F}_q^m$ a solution $\mathbf{y}$ for $\mathcal{P}(\mathbf{y}) = \mathbf{x}$ can be found by first solving $\mathcal{F}(\mathbf{y}') = \mathbf{x}$ for $\mathbf{y}'$ and then computing $\mathbf{y} = \mathcal{T}^{-1}(\mathbf{y}')$. The system $\mathcal{F}(\mathbf{y}') = \mathbf{x}$ can be solved efficiently by randomly choosing the values of the vinegar variables. If we substitute these values in the equations the remaining system only contains linear equations, because every quadratic term contains at least one vinegar variable and thus turns into a linear or constant term after substitution. The remaining linear system can be solved using linear algebra. In the event that there are no solutions we can simply try again with a different choice for the vinegar variables.

The trapdoor function is then combined with a collision resistant hash function $\mathcal{H} : \{0,1\}^* \to \mathbb{F}_q^m$ into a signature scheme using the standard hash-and-sign paradigm. The resulting key generation, signature generation and verification algorithms of the UOV signature scheme are described in Algorithms 1, 2 and 3.

```
Algorithm UOVGenerateKeys
input: Random bits to generate $\mathcal{F}$ and $\mathcal{T}$
output: $\mathcal{P}$ — A public key
        $(\mathcal{F}, \mathcal{T})$ — A corresponding secret key
1: $\mathcal{F} \leftarrow$ A randomly chosen UOV system
2: $\mathcal{T} \leftarrow$ A randomly chosen linear map $\mathbb{F}_q^n \to \mathbb{F}_q^n$
3: $\mathcal{P} \leftarrow \mathcal{F} \circ \mathcal{T}$
4: return $\mathcal{P}$ and $(\mathcal{F}, \mathcal{T})$
```

Algorithm 1. The UOV key pair generation algorithm

3.2 Attacks against UOV

Direct attack. This attack tries to forge a signature s for a message M by solving the polynomial system $\mathcal{P}(s) = \mathcal{H}(M)$. An attacker can use the trick of Thomae and Wolf [27] to reduce this to finding a solution of a polynomial system with $m + 1 - \lfloor n/m \rfloor$ equations. The best known algorithms to solve this problem use the hybrid approach [5] which was briefly described in Sect. 2.

ows the fact that in order for the signature scheme to be secure, the number of hen (vinegar) variables should be larger than the number of rooster (oil) variables. Nevertheless, we will stick to the traditional naming of oil and vinegar variables.

```
Algorithm UOVSign
input: $(\mathcal{F}, \mathcal{T})$ — A secret key
       $M$ — A message to sign
output: $\mathbf{s}$ — A signature for the message $M$
 1: $\mathbf{h} \leftarrow \mathcal{H}(M)$
 2: while No solution $\mathbf{s}'$ to the system $\mathcal{F}(\mathbf{s}') = \mathbf{h}$ is found do
 3:     Assign random values to the first $v$ entries of $\mathbf{s}'$
 4:     Substitute these values into $\mathcal{F}(\mathbf{s}') = \mathbf{h}$ to get a linear system $L(\mathbf{o}) = \mathbf{h}$.
 5:     if $L(\mathbf{o}) = h$ has solutions then
 6:         Calculate an assignment $\mathbf{o}$ to the oil variables such that $L(\mathbf{o}) = \mathbf{h}$
 7:         Assign the entries of $\mathbf{o}$ to the last $m$ entries of $\mathbf{s}'$
 8:     end if
 9: end while
10: $\mathbf{s} \leftarrow \mathcal{T}^{-1}(\mathbf{s}')$
11: return $\mathbf{s}$
```

Algorithm 2. The UOV signature generation algorithm

```
Algorithm UOVVerify
input: $\mathcal{P}$ — A public key
       $M$ — A message
       $\mathbf{s}$ — A candidate–signature
output: True if $\mathbf{s}$ is a valid signature for $M$, False otherwise
 1: $\mathbf{h} \leftarrow \mathcal{H}(M)$
 2: $\mathbf{h}' \leftarrow \mathcal{P}(\mathbf{s})$
 3: if $\mathbf{h} = \mathbf{h}'$ then
 4:     return True
 5: else
 6:     return False
 7: end if
```

Algorithm 3. The UOV signature verification algorithm

Empirically, the systems that have to be solved behave like semi-regular systems [12], therefore we can calculate the degree of regularity and use this to estimate the complexity of the hybrid approach. Petzoldt [22] uses a similar method to estimate the complexity of a direct attack against UOV, the only difference being that we have used an updated estimate of the complexity of F_5 [6]. In Petzoldt's thesis it was shown that the estimated complexity of a direct attack agrees very well with the measured complexity of a direct attack against small instances of UOV. These experiments justify ignoring the big-O notation in formula (1) and treating the formula as an estimate for the concrete hardness of the hybrid approach.

Example 1. We will estimate the complexity of a direct attack against UOV with the parameter set $(q = 31, m = 52, v = 104)$; this set is proposed in [22] as a set that achieves 128-bit security. Using the trick of Thomae et al. we can reduce finding a solution to this underdetermined system to finding a solution of a determined system with $52 + 1 - \lfloor (52 + 104)/52 \rfloor = 50$ equations. We assume this system to be semi-regular. If we fix k extra variables the degree of regularity is equal to the degree of the first term in the power series of

$$S_{50,50-k}(x) = \frac{(1 - x^2)^{50}}{(1 - x)^{50-k}}$$

which has a non-positive coefficient. For $k = 0$ we have $S_{50,50}(x) = (1 + x)^{50}$, so the degree of regularity is 51. For $k = 1$ we have

$$S_{50,49}(x) = 1 + 49x + 1175x^3 + \cdots + 4861946401452x^{25} - 4861946401452x^{26} + O(x^{27}),$$

where all the omitted terms have positive coefficients, so the degree of regularity is 26. We can now use (1) to estimate the complexity of the hybrid approach. We prefer to err on the side of caution, so we have chosen $\omega = 2$ for the value of the linear algebra constant. For k equal to 0 and 1 this is equal to

$$\binom{50 + 51}{51}^2 \approx 2^{194.7} \qquad \text{and} \qquad 31\binom{50 - 1 + 26}{26}^2 \approx 2^{137.8}$$

respectively. Continuing this for higher values of k we eventually see that the optimal value of k is 6, the corresponding degree of regularity is 16 and the complexity of the direct attack is $2^{123.9}$.

In the example we concluded that the complexity of the attack is less than 2^{128} which was supposed to be the security level of the parameter set $(q = 31, m = 52, v = 104)$ according to [22]. Even though we have used roughly the same method of estimating the complexity as the method used by Petzoldt [22] we arrive at a slightly different value because we have used a tighter bound on the complexity of F_5 coming from an improved analysis of the hybrid approach [6].

With this method we can calculate the minimal number of equations that is needed in a determined semi-regular system in order to guarantee that the complexity of finding a solution is larger than a targeted security level. For quantum attackers, we can follow the same method with (2) instead of (1) for estimating the complexity of the hybrid approach. The result of these calculations for the security levels of 2^{128} and 2^{256} for different finite fields of size up to $q = 2^{100}$ are plotted in Fig. 1.

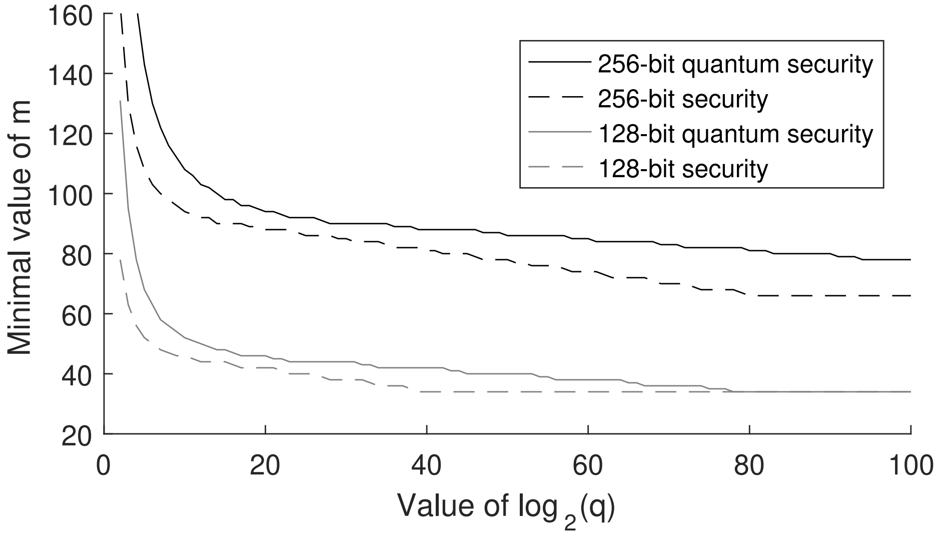


Fig. 1. The minimal sizes of determined semi-regular systems to reach 128-bit security and 256-bit security for different finite fields.

UOV attack. Patarin [20] suggested in the original version of the Oil and Vinegar scheme to choose the same number of vinegar and oil variables, or $v = m$. This choice was cryptanalyzed by Kipnis and Shamir [16]: they showed that an attacker can find the inverse image of the oil variables under the map $\mathcal{T}$. This is enough information to find an equivalent secret key, so this breaks the scheme. This approach generalizes for the case $v > m$; the complexity then increases to $O(q^{v-m}n^4)$ [15] and is thus exponential in $v - m$. Typically one chooses $v = 2m$ or $v = 3m$ to preclude the UOV attack.

UOV reconciliation attack. Similar to the UOV attack, the UOV reconciliation attack proposed by Ding et al. [9] tries to find an equivalent secret key. We present a brief summary. In this section we will make a distinction between m, the number of polynomials in the public and private system, and o, the number of oil variables. In the UOV signature scheme these numbers are the same, which explains why we did not need to make this distinction before. It turns out that for a public key $\mathcal{P}$ there exists with a very high probability a private key $(\mathcal{F}, \mathcal{T})$ such that the matrix representation of $\mathcal{T}$ is of the form

$$\mathbf{M}_{\mathcal{T}} = \begin{pmatrix} \mathbf{I}_v & \mathbf{T} \\ 0 & \mathbf{I}_o \end{pmatrix}.$$

This means that an attacker only has to find the $v \times o$ matrix $\mathbf{T}$ to get an equivalent key. The UOV reconciliation attack tries to find $\mathbf{T}$ algebraically by solving a quadratic system. If the choice of $\mathcal{T}$ is correct (i.e. there exists a private key of the form $(\mathcal{F}, \mathcal{T})$, then we have that the matrix representation $\mathbf{P}_i$ of the quadratic part of each polynomial in the public key satisfies for all $1 \leq i \leq m$

$$\begin{pmatrix} *_{v\times v} & *_{o\times v} \\ *_{v\times o} & 0_{o\times o} \end{pmatrix} = \begin{pmatrix} \mathbf{I}_v & 0 \\ -\mathbf{T} & \mathbf{I}_o \end{pmatrix} \mathbf{P}_i \begin{pmatrix} \mathbf{I}_v & -\mathbf{T} \\ 0 & \mathbf{I}_o \end{pmatrix}. \tag{3}$$

The condition that the lower right $o \times o$ submatrices of the private system consist of zeroes give quadratic equations in the entries of $\mathbf{T}$. It looks like we have o^2 equations for each component, but since the matrix representations are only defined up to the addition of a skew symmetric matrix this gives only $o(o+1)/2$ equations per component. In total we have a system of $mo(o+1)/2$ equations in vo variables. The reconciliation attack tries to recover $\mathbf{T}$ by solving this system of equations.

The reconciliation system has a structure that makes it much easier to solve compared to a random system of the same size. In fact, Ding et al. argue that the complexity of this attack for UOV variants with $v \leq m$ (like Rainbow and TTS) is the same as the complexity of solving a system of m equations in v variables [9].

In the case $v \geq m$ the complexity of the attack is more difficult to estimate, but we can formulate a lower bound to the complexity of the attack. The reconciliation system has $mo(o+1)/2$ equations in ov variables. For all parameter choices of UOV this is a heavily overdetermined system, so it should not be a surprise that there is only one matrix $\mathbf{T}$ that satisfies (3). Computer experiments have shown that there is a unique solution for $\mathbf{T}$ as soon as the number of equations of the reconciliation system exceeds the number of variables. Let $\text{Rec}[v, o, m]$ denote the complexity of a key reconciliation attack against a UOV public system with v vinegar variables, o oil variables and m polynomials in the public key. Increasing m only makes the reconciliation attack easier. Indeed, increasing the number of equations can only make the attack easier, because an attacker could just ignore the extra equations and still find the same unique solution. In other words, if $m < m'$, then we have $\text{Rec}[v, o, m] \geq \text{Rec}[v, o, m']$, provided that $mo(o+1)/2 > ov$, which is the case for all good UOV parameter choices.

We can now derive a lower bound on the complexity of a reconciliation attack when $v > m = o$. According to the above observation, we can increase m, the number of equations, until it matches the number of vinegar variables v, and this would make solving the system easier, i.e. we have

$$\text{Rec}[v, m, m] \geq \text{Rec}[v, m, v]\,. \tag{4}$$

We can now use the argument of Ding et al. which says that when $m \geq v$, the complexity of the reconciliation attack is equal to the complexity of solving a system of m quadratic equations in v variables, so $\text{Rec}[v, m, v]$ is equal to the complexity of solving a system of v quadratic equations in v variables.

We conclude that a UOV reconciliation attack on a UOV system with m equations and $v \geq m$ vinegar variables is at least as difficult as solving a system of v quadratic variables in v equations, but it is expected to be more difficult, because a lot of hardness is lost in the inequality (4). In particular, the reconciliation attack is less effective against the UOV scheme than attacking the system $\mathcal{P}(s) = \mathcal{H}(M)$ directly.

Quantum attacks. There are no known specialized quantum algorithms that solve multivariate quadratic equations. However, as described in Sect. 2, Grover's algorithms can be used to speed up the exhaustive search part of hybrid solution finding algorithms. This quantum version of the hybrid approach algorithm can be used to speed up a direct attack and a reconciliation attack.

Grover search could be used to speed up the UOV attack from $O(q^{v-m}n^4)$ to $O(q^{\frac{v-m}{2}}n^4)$. This requires repeatedly running an algorithm that calculates the common eigenspaces of a set of matrices and checks whether any of these eigenspaces lies within the oil subspace in superposition. In comparison with the classical algorithm this has the disadvantage that it cannot be parallelized without a significant amount of overhead.

4 Improving UOV

In [22] Petzoldt presented a new method to reduce the public key size of UOV by roughly a factor 8. The key generation algorithm was adapted to make it possible to choose a large part of the public key. One can generate this part with a pseudo-random number generator and replace a large part of the public key by a seed. Also, it is possible to choose part of the public key in such a way such that signatures can be verified faster [21].

Usually, during key generation, a UOV system $\mathcal{F}$ and an invertible linear map $\mathcal{T}$ are chosen randomly, and then $\mathcal{P}$ is determined as $\mathcal{P} = \mathcal{F} \circ \mathcal{T}$. With this strategy we have full control over $\mathcal{F}$, but no control over the public key $\mathcal{P}$. Instead, Petzoldt proposed to first pick $\mathcal{T}$ and $v(v+1)/2 + mv$ coefficients of each polynomial of $\mathcal{P}$. Then we solve the system $\mathcal{P} = \mathcal{F} \circ \mathcal{T}$ to find the coefficients of $\mathcal{F}$, and the remaining coefficients of $\mathcal{P}$. This is a linear system of equations, so this can happen efficiently. With a small probability this system does not have any solutions, but in that case we can simply try again with a different choice of $\mathcal{T}$. For the details of this method we refer to [22].

With this approach the public key size is decreased with $m(v(v+1)/2+mv)$ field elements, at the negligible cost of including the seed for the random number generator. The public key size is now $m^2(m+1)/2 \log_2(q) + |\text{seed}|$. Table 1 shows that this method drastically reduces the size of the public key. However, the public key remains much larger than the signature schemes that are in use today such as RSA [25] and DSA [17], which typically stay well under 1 kB. Note that if Petzoldt's method is used, the size of the public key is independent of v, the number of vinegar variables.

Table 1. The effect of Petzoldt's method on the public key size

security level	q	(m, v)	public key (kB)	public key with Petzoldt's method (kB)
100-bit	2^8	(36,72)	207	23
128-bit	2^8	(47,94)	460	52
192-bit	2^8	(72,144)	1648	185
256-bit	2^8	(98,196)	4150	464

5 Lifting $\mathcal{P}$ to an extension field

In this section we will work with UOV over a finite field $\mathbb{F}_{2^r}$ of characteristic 2. The parameter r is quite important for the security of the scheme, the signature size and key sizes. It can be seen in Fig. 1 that by choosing a larger value of r we can put a smaller number of equations in the system and still reach the same level of security. Since the number of field elements in the public key and secret key is $O(m^3)$ it is desirable to have a small value of m. However, since it costs r bits to store a field element r should not be too big either. We must make a trade-off between large r and large m. In this section we propose a scheme that gets some security benefits of a high value of r, but has a public and private key with coefficients in $\mathbb{F}_2$, greatly reducing the key sizes.

5.1 Description of the new scheme

As usual, the public key of the scheme represents a quadratic system over $\mathbb{F}_{2^r}$, given by

$$\mathcal{P} = \mathcal{F} \circ \mathcal{T} .$$

When we want to sign a message m we use a hash function to generate a digest of mr bits which represents a vector $\mathbf{h}$ of m elements of $\mathbb{F}_{2^r}$. Then we use the knowledge of the private key to solve the system $\mathcal{P}(\mathbf{s}) = \mathbf{h}$ to get a valid signature $\mathbf{s}$. However, the difference with standard UOV is that we now choose all the coefficients of $\mathcal{F}, \mathcal{P}$ and $\mathcal{T}$ in $\mathbb{F}_2$. Therefore the key generation process is identical to the key generation process of a regular UOV scheme over $\mathbb{F}_2$. In particular, we can use the approach of Petzoldt [22] as explained in Sect. 4 to reduce the size of the public key. Contrary to the key generation, the signature generation and verification still happen over the field $\mathbb{F}_{2^r}$ as usual.

To summarize, we simply take a key pair of the UOV scheme over $\mathbb{F}_2$, and use it as a key pair for the UOV scheme over $\mathbb{F}_{2^r}$. The public key is thus approximately a factor r smaller than if we were to use the regular UOV scheme over $\mathbb{F}_{2^r}$ since we only use one bit to represent each coefficient instead of r bits. Furthermore, we can now choose r to be much larger than what would otherwise its optimal value. This in turn allows for a smaller value of m (See Fig. 1), reducing the public key size even more.

The public key consists of a seed for a pseudorandom number generator and the part of the public map which cannot be generated. The total size of the public key is therefore

$$|\text{seed}| + \frac{m^2(m+1)}{2} \text{ bits.}$$

Storing the private maps $\mathcal{F}$ and $\mathcal{T}$ would take

$$m\frac{v(v+1)}{2} + m^2 v \text{ bits and } n^2 \text{ bits}$$

respectively, but they do not need to be stored, because they can be calculated using the key generation algorithm each time they are needed. A signature consists of $n = m + v$ elements of $\mathbb{F}_{2^r}$, so the size of the signature is nr bits.

Remark 2. Though we have presented this scheme with a finite field of characteristic 2 and with the subfield $\mathbb{F}_2 \subset \mathbb{F}_{2^r}$, it is easy to see that we can use this scheme with any field extension of finite fields $K \subset K'$. In such a scenario we generate a key pair with coefficients in the small field K, and the signing and verifying is done with elements of the big field K'.

5.2 Security analysis of the new scheme

Direct attack. This attack tries to forge a signature for a certain message M by trying to find a solution $s \in \mathbb{F}_{2^r}^n$ for the system $\mathcal{F}(s) = \mathcal{H}(M)$. The best known methods for this use the hybrid approach as described in Sect. 2.

For a direct attack against the new scheme all the coefficients of the system that needs to be solved lie in $\mathbb{F}_2$, except those of the constant terms, because those coefficients come from the message digest. We claim that this does not significantly reduce the hardness of finding solutions relative to the case where the coefficients are generic elements of $\mathbb{F}_{2^r}$. It has been noticed by Faugère and Perret [12] that the polynomial systems that result from fixing $\approx v$ variables in a UOV system behave like semi-regular systems. The degree of regularity of a quadratic semi-regular system is given by the degree of the first term in the power series of

$$\frac{(1-x^2)^m}{(1-x)^n}$$

with a non-positive coefficient. In particular the degree of regularity does not depend on q for semi-regular systems. Hence, the degree of regularity for a direct attack against the modified UOV scheme is identical to the degree of regularity of an attack against the regular UOV scheme. Therefore a Gröbner basis computation against the modified scheme is not significantly more efficient than a Gröbner basis computation against regular UOV with the same parameters. This argument is confirmed by the experimental data in Table 2. There we see that a direct attack is slightly faster against the modified scheme than against the original UOV scheme, but only by a small constant factor. Even though the Gröbner basis is computed over $\mathbb{F}_{2^r}$, the largest part of the arithmetic only involves the field elements 0 and 1, so the arithmetic is faster than with generic elements of $\mathbb{F}_{2^r}$. This is where the difference observed in Table 2 comes from. If we do the same experiment with a smaller extension field such as $\mathbb{F}_{2^8}$ there is no observed difference between the running time of a direct attack against a regular UOV scheme and our modified scheme.

Remark 3. In a direct attack one fixes $\approx v$ variables randomly to make the system a slightly overdetermined system. In our experiments we have fixed these variables to values in $\mathbb{F}_2$ to make sure that we do not introduce linear terms with coefficients in $\mathbb{F}_{2^r}$ instead of $\mathbb{F}_2$ in the case of the modified UOV scheme.

Table 2. Running time of a direct attack against the regular UOV scheme over $\mathbb{F}_{2^{64}}$ and the modified UOV scheme, with the MAGMA v2.22-10 implementation of the F4 algorithm. We did not implement the method of Thomae and Wolf [27].

(m,v)	Regular UOV (s)	Lifted UOV (s)	difference
(7,35)	0.43	0.21	-52%
(8,40)	1.56	0.76	-51%
(9,45)	7.00	3.21	-54%
(10,50)	33.50	17.44	-48%
(11,55)	132.88	76.60	-42%
(12,60)	828.31	588.33	-29%

Remark 4. It might seem tempting to decompose the equations over $\mathbb{F}_{2^r}$ into equations over $\mathbb{F}_2$ to make a direct attack more efficient. This decomposition is done by fixing some basis $\beta_1, \cdots, \beta_r$ of $\mathbb{F}_{2^r}$ over $\mathbb{F}_2$ and replacing each variable x_i by $\sum_{j=1}^{r} \hat{x}_{i,j}\beta_j$, where the $\hat{x}_{i,j}$ are nr new variables in $\mathbb{F}_2$. Each equation of the original system is then decomposed into r equations, resulting in a total of mr equations in nm variables over $\mathbb{F}_2$. The problem with this approach is that the number of equations and variables is increased by the factor r, which makes the naive approach of solving the decomposed system with a generic boolean solver hopelessly slow. However, the decomposed system has a specific structure which could potentially be exploited to solve the system more efficiently. We investigated this possibility, but we we were not able to make any progress. It

should be pointed out that this idea does not only apply to our scheme, but to any multivariate cryptosystem over a field of non-prime order. Still, no such attacks are reported in literature. One could say that the idea of decomposing a system to make it easier to solve is not very promising because in big-field schemes such as Gui [24] and medium-field schemes such as HMFEv [23] the systems are decomposed with the objective of making them *harder* to solve for an attacker.

Key recovery attacks. In contrast to a direct attack, the modified scheme is more vulnerable to a key recovery attack. Since the key pair used in the Lifted UOV scheme is identical to the key pair of regular UOV over the field $\mathbb{F}_2$ it is clear that a key recovery attack against the Lifted UOV scheme is equivalent to a key recovery attack against a regular UOV scheme over $\mathbb{F}_2$, which is much easier than a key recovery attack against UOV over $\mathbb{F}_{2^r}$. Luckily, key recovery attacks against UOV have been investigated ever since the invention of the oil and vinegar scheme in 1997 [20], so it is well understood which attacks are possible (see Sect. 3.2) and what the complexities of these attacks are. It is also clear that we can make key recovery attacks harder by increasing the number of vinegar variables.

The UOV attack attempts to recover an equivalent private key by searching for the oil subspace. This attack has complexity $q^{v-m-1} \cdot n^4$. Since a UOV attack on the Lifted UOV scheme is equivalent to a UOV attack over $\mathbb{F}_2$, we have that the complexity of a UOV attack against the Lifted UOV scheme is $2^{v-m-1} \cdot n^4$.

The reconciliation attack against the lifted UOV scheme is equivalent to the UOV reconciliation attack against UOV over the field $\mathbb{F}_2$. A lower bound on the complexity of this attack is given by the complexity of solving a quadratic system of v variables and v equations over $\mathbb{F}_2$, but we expect the problem to be harder. There exists specialized algorithms for solving polynomial systems over $\mathbb{F}_2$ that are more efficient than the generic hybrid approach. One method is a smart exhaustive search, which requires approximately $log_2(n)2^{n+2}$ bit operations [7]. The BooleanSolve algorithm [3] combines an exhaustive search with sparse linear algebra to achieve a complexity of $O(2^{0.792n})$. However the method only becomes faster than the exhaustive search method when $n > 200$. Recently, Joux et al. proposed a new algorithm that was able to solve a boolean system of 146 quadratic equations in 73 variables in one day [14]. The algorithm beats the exhaustive search algorithm, even for small systems. The complexity of this algorithm is still under investigation, but a rough estimate based on the reported experiments suggests that it scales like $2^{\alpha n}$ with α between 0.8 and 0.85 and with a small constant factor. For choosing the parameters of our signature scheme, we have assumed that a determined system of n quadratic boolean equations provides $0.75n$ bits of security, even though this is likely to seriously overestimate the capabilities of the state of the art algorithms. Quantum attackers can use Grover search to solve systems over $\mathbb{F}_2$ with complexity $O(2^{n/2})$.

5.3 Choice of parameters

For convenience and efficiency we will work with binary finite fields whose elements are represented by a number of bits that is a multiple of 16, i.e. the finite fields we want to use are $\mathbb{F}_{2^{16}}, \mathbb{F}_{2^{32}}, \mathbb{F}_{2^{48}}$ and so on.

When designing a signature scheme of security level l, we choose a finite field that is large enough such that the minimal number of equations in a determined regular system that is needed to reach the security level l is minimized. Figure 1 shows that for 128-bit and 256-bit security the chosen fields are $\mathbb{F}_{2^{48}}$ and $\mathbb{F}_{2^{80}}$ respectively, and the minimal number of equations is 34 and 66 respectively or 40 and 81 when considering quantum attacks. For 100-bit and 192-bit security the chosen fields are $\mathbb{F}_{2^{32}}$ and $\mathbb{F}_{2^{64}}$, and the minimal number of equations is 27 and 50 for classical attackers or 33 and 60 for quantum attackers.

We now consider the constraints on the parameters due to the different attacks against our scheme. In order to be safe against a direct attack we require

$$m - \lfloor v/m \rfloor \geq m_{min} \,,$$

with m_{min} equal to $27, 34, 50$ or 66 if the desired security level is 100 bits, 128 bits, 192 bits, or 256 bits respectively. For quantum attackers m_{min} is equal to $33, 40, 60$ and 81 respectively. In order to be safe against the UOV attack we require

$$2^{v-m-1} n^4 > 2^l \qquad \text{or} \qquad 2^{(v-m-1)/2} n^4 > 2^l \,,$$

depending on whether we want l bits of security against classical, or quantum adversaries. To be secure against the UOV reconciliation attack it suffices that an attacker cannot solve a determined system with v equations over $\mathbb{F}_2$. Therefore it suffices to have

$$2^{0.75v} > 2^l \qquad \text{or} \qquad 2^{v/2} > 2^l$$

for classical and quantum attackers respectively. The parameter sets displayed in Table 3 satisfy all the constraints for the targeted security level and minimize the size of the public key, i.e. they minimize m. In the last column of the table, the bit complexity of the best known classical attack against the parameter set is calculated. For all the proposed parameters the best known classical attack is a direct Groebner basis attack.

Table 3. Parameter choices and corresponding public key and signature sizes for different security levels

security level		(r, m, v)	|pk| (kB)	|sig| (kB)	classical security
100-bit	classical	(32,31,134)	1.9	0.6	
	quantum	(32,37,200)	3.2	0.9	115 bit
128-bit	classical	(48,38,171)	3.4	1.2	
	quantum	(48,45,256)	5.7	1.8	153 bit
192-bit	classical	(64,54,256)	9.8	2.4	
	quantum	(64,65,384)	17.0	3.5	224 bit
256-bit	classical	(80,70,341)	21.2	4.0	
	quantum	(80,87,526)	40.7	6.0	296 bit

5.4 Trade-off

In comparison to regular UOV, Lifted UOV has much smaller public keys, but also larger signatures. In the discussion above, we have chosen the parameter r very large in order to minimize the size of the public key, without considering the size of the signatures. It is possible to make a trade-off between the size of the public key and the size of the signature by choosing a smaller value of r. Having a smaller value of r requires a larger value of m to reach the same security level, resulting in a larger public key, but since the signature consists of n elements of $\mathbb{F}_{2^r}$ it also leads to smaller signatures. Figure 2 compares public key sizes and signature sizes of the Lifted UOV scheme with different values of the parameter r with some other MQ signature schemes [22], the lattice-based signature scheme BLISS-II [10] and SPHINCS, a hash-based signature scheme [4]. Note that even though the MQ schemes UOVRand and RainbowLRS2 claim to provide 128-bit of post-quantum security, their parameters are not chosen to resist quantum attacks on the MQ problem or quantum versions of the UOV attack. So we are not comparing schemes with the same security level. Ignoring quantum attacks, the Lifted UOV signature scheme with $r = 48$ in the comparison achieves 153 bits of security.

Example 2. For some application on a low-cost device it might be desirable to have a signature scheme that provides 128 bits of post-quantum security with minimal signature sizes subject to the condition that the public key is smaller than, say, 10 kB. If we choose the parameters as in the discussion above, we would have a public key of 5.7 kB and signatures of 1.8 kB. However, we can do better by choosing $r = 12$. The lowest values of m and v providing 128 bits of security are then $m = 54$ and $v = 256$. This leads to a public key of 9.8 kB (< 10kB) and a signature of 0.45 kB.

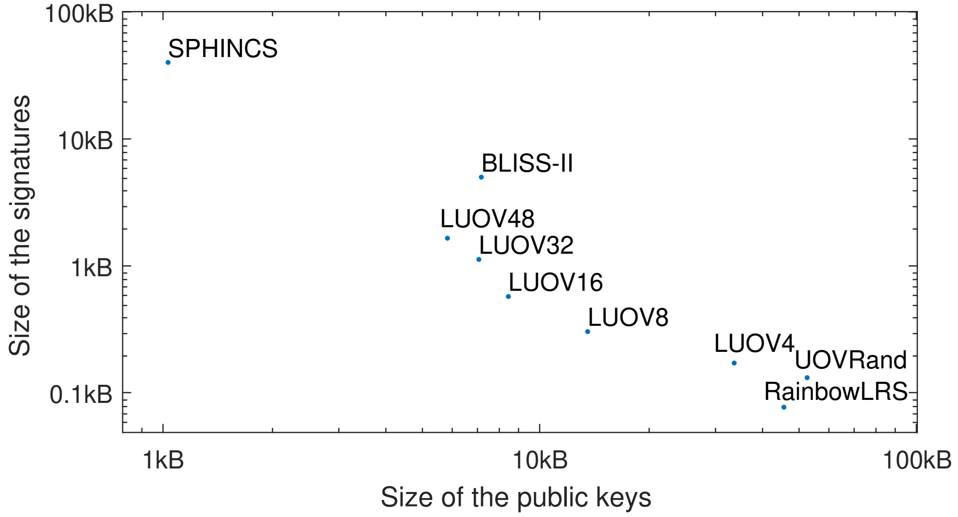


Fig. 2. Comparison of different signature schemes providing 128 bits of post-quantum security.

6 Implementation and results

We developed an ANSI C implementation of the Lifted UOV signature scheme. The large fields are implemented as extension fields of $\mathbb{F}_{2^{16}}$ and the arithmetic in $\mathbb{F}_{2^{16}}$ is done using log tables. We have a table that maps each nonzero element x to the number y such that $x = a^y$, where a is some generator of the group $\mathbb{F}_{2^{16}}^{\times}$. Conversely, we also have a table that maps a number y to the element a^y. Multiplication in $\mathbb{F}_{2^{16}}$ is then computed with three table lookups and an addition modulo $2^{16} - 1$. Note that this approach could make our implementation vulnerable to cache timing attacks. Newer CPUs support the

CLMUL instruction set which could be used to perform the field arithmetic efficiently without the need for lookup tables, eliminating the possibility of this attack. Two field elements are added using a XOR operation. During the key generation phase we only use elements of $\mathbb{F}_2$, so we have used bit slicing whenever possible to speed up the algorithm. The running times of the key generation, signature generation and the verification algorithms are displayed in Table 4.

Please note that the implementation uses naive implementations of matrix multiplication, polynomial multiplication and Gaussian reduction, and the code was not heavily optimized. Therefore, it can be expected that the running times reported in Table 4 are nowhere near optimal. Some techniques that can speed up the code very significantly include writing cache friendly code, using parallelization and using Karatsuba's algorithms for the field arithmetic. Moreover, it is possible to use a method of Petzoldt to structure part of the public key in such a way that the verification algorithm is faster [22]. In order to avoid storing the large private key, part of the key generation algorithm is run each time a signature is generated to generate the private key. If a batch of messages is signed together this step only has to happen once. Alternatively, if storing the private key is not an issue, this part can be omitted altogether to speed up the signing algorithm significantly.

Table 4. Running times for the key generation, signing and verification algorithms on a single thread on an Intel®Core™ i7-4710MQ CPU at 2.5 GHz

security level		key gen (ms)	sig gen (ms)	verification (ms)
100-bit	classical	4	6	3
	quantum	13	16	7
128-bit	classical	10	14	7
	quantum	26	34	15
192-bit	classical	32	46	21
	quantum	148	156	54
256-bit	classical	125	149	55
	quantum	366	410	144

7 Conclusion

The simple idea of lifting a UOV key pair from $\mathbb{F}_2$ to an extension field $\mathbb{F}_{2^r}$ increases the security against direct attacks without affecting the size of the public key. At the same time, thanks to the method of Petzoldt, we can increase the number of vinegar variables to protect against key recovery attacks without increasing the size of the public key. These two ideas come together to create a secure signature scheme whose public key is an order of magnitude smaller than other MQ signature schemes, with slightly larger signatures. The signature scheme is very competitive with other post-quantum signature schemes. By choosing the parameter r it is possible to make a trade-off between larger public keys and smaller signatures or vice versa. We developed a rudimentary ANSI C implementation of the Lifted UOV signature scheme which shows that key generation, signing and verification takes only a few milliseconds for 100-bit security instantiations of the scheme and up to a few hundred milliseconds for 256-bit security instantiations. However it is very likely that these times can be improved significantly with an optimized implementation.

The idea of lifting keys to a large extension field can be applied to any MQ signature scheme, but it might not always be useful to produce smaller public keys. We believe that the idea could be used to improve the Rainbow signature scheme, but not HFE or C^*. This is because the public keys of signature schemes such as HFE and C^* are not semi-regular maps [11] and have a much smaller degree of regularity than random maps of the same dimensions. This means that guessing a few variables does not necessarily reduce the degree of regularity, like it does in the case of semi-regular systems. This makes the hybrid approach unsuitable for attacking these systems, since solving the system with one big Gröbner basis computation is more efficient. Therefore there is no point in lifting the system to a larger field, because the complexity of a Gröbner basis computation is largely independent of the size of the finite field.

Acknowledgements This work was supported in part by the Research Council KU Leuven: C16/15/058. In addition, this work was supported by the European Commission through the Horizon 2020 research and innovation programme under grant agreement No. H2020-ICT-2014-644371 WITDOM and H2020-ICT-2014-645622 PQCRYPTO.

References

1. PQCRYPTO ICT-645622 (2015), http://pqcrypto.eu.org/
2. Bardet, M.: Étude des systèmes algébriques surdéterminés. Applications aux codes correcteurs et à la cryptographie. Ph.D. thesis, Université Pierre et Marie Curie-Paris VI (2004)

3. Bardet, M., Faugère, J.C., Salvy, B., Spaenlehauer, P.J.: On the complexity of solving quadratic Boolean systems. Journal of Complexity 29(1), 53–75 (2013)
4. Bernstein, D.J., Hopwood, D., Hülsing, A., Lange, T., Niederhagen, R., Papachristodoulou, L., Schneider, M., Schwabe, P., Wilcox-O'Hearn, Z.: SPHINCS: practical stateless hash-based signatures. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 368–397. Springer (2015)
5. Bettale, L., Faugere, J.C., Perret, L.: Hybrid approach for solving multivariate systems over finite fields. Journal of Mathematical Cryptology 3(3), 177–197 (2009)
6. Bettale, L., Faugère, J.C., Perret, L.: Solving polynomial systems over finite fields: Improved analysis of the hybrid approach. In: Proceedings of the 37th International Symposium on Symbolic and Algebraic Computation. pp. 67–74. ACM (2012)
7. Bouillaguet, C., Chen, H.C., Cheng, C.M., Chou, T., Niederhagen, R., Shamir, A., Yang, B.Y.: Fast exhaustive search for polynomial systems in $\mathbb{F}_2$. In: International Workshop on Cryptographic Hardware and Embedded Systems. pp. 203–218. Springer (2010)
8. Diem, C.: The XL-algorithm and a conjecture from commutative algebra. In: Asiacrypt. vol. 4, pp. 338–353. Springer (2004)
9. Ding, J., Yang, B.Y., Chen, C.H.O., Chen, M.S., Cheng, C.M.: New differential-algebraic attacks and reparametrization of Rainbow. In: International Conference on Applied Cryptography and Network Security. pp. 242–257. Springer (2008)
10. Ducas, L., Durmus, A., Lepoint, T., Lyubashevsky, V.: Lattice signatures and bimodal gaussians. In: Advances in Cryptology–CRYPTO 2013, pp. 40–56. Springer (2013)
11. Faugere, J.C., Joux, A.: Algebraic cryptanalysis of hidden field equation (HFE) cryptosystems using Gröbner bases. In: Annual International Cryptology Conference. pp. 44–60. Springer (2003)
12. Faugère, J.C., Perret, L.: On the security of UOV. IACR Cryptology ePrint Archive 2009, 483 (2009)
13. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing. pp. 212–219. ACM (1996)
14. Joux, A., Vitse, V.: A crossbred algorithm for solving Boolean polynomial systems. IACR Cryptology ePrint Archive 2017, 372 (2017)
15. Kipnis, A., Patarin, J., Goubin, L.: Unbalanced oil and vinegar signature schemes. In: International Conference on the Theory and Applications of Cryptographic Techniques. pp. 206–222. Springer (1999)
16. Kipnis, A., Shamir, A.: Cryptanalysis of the oil and vinegar signature scheme. In: Annual International Cryptology Conference. pp. 257–266. Springer (1998)
17. Kravitz, D.W.: Digital signature algorithm (Jul 27 1993), US Patent 5,231,668
18. Michael, R.G., David, S.J.: Computers and intractability: a guide to the theory of NP-completeness. WH Free. Co., San Fr (1979)
19. National Institute for Standards and Technology (NIST): Post-quantum crypto standardization (2016), http://csrc.nist.gov/groups/ST/post-quantum-crypto/

20. Patarin, J.: The oil and vinegar signature scheme. In: Dagstuhl Workshop on Cryptography1997 (1997)
21. Petzoldt, A.: Hybrid approach for the fast verification for improved versions of the UOV and Rainbow signature schemes. IACR Cryptology ePrint Archive 2013, 315 (2013)
22. Petzoldt, A.: Selecting and Reducing Key Sizes for Multivariate Cryptography. Ph.D. thesis, TU Darmstadt (Jul 2013), referenten: Professor Dr. Johannes Buchmann, Professor Jintai Ding, Ph.D.
23. Petzoldt, A., Chen, M.S., Ding, J., Yang, B.Y.: Hmfev-an efficient multivariate signature scheme. In: International Workshop on Post-Quantum Cryptography. pp. 205–223. Springer (2017)
24. Petzoldt, A., Chen, M.S., Yang, B.Y., Tao, C., Ding, J.: Design principles for hfev-based multivariate signature schemes. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 311–334. Springer (2015)
25. Rivest, R.L., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM 21(2), 120–126 (1978)
26. Shor, P.W.: Polynomial time algorithms for discrete logarithms and factoring on a quantum computer. In: International Algorithmic Number Theory Symposium. vol. 877, pp. 289–289. Springer (1994)
27. Thomae, E., Wolf, C.: Solving underdetermined systems of multivariate quadratic equations revisited. In: International Workshop on Public Key Cryptography. pp. 156–171. Springer (2012)
28. Zalka, C.: Grover's quantum searching algorithm is optimal. Physical Review A 60(4), 2746 (1999)

Chapter 14

MUDFISH and SUSHSYFISH

Overall, the proposed protocols look really "fishy" to me.

– anonymous reviewer #2

Publication Data

Beullens W. Sigma Protocols for MQ, PKP and SIS, and Fishy Signature Schemes. In: Advances in Cryptology – EUROCRYPT 2020. Lecture Notes in Computer Science, vol 12107, pages 183-211, Springer, 2020.

Sigma protocols for MQ, PKP and SIS, and fishy signature schemes

Ward Beullens[1]

imec-COSIC KU Leuven,
Kasteelpark Arenberg 10 - bus 2452, 3001 Heverlee, Belgium
Ward.Beullens@esat.kuleuven.be

Abstract. This work presents sigma protocols to prove knowledge of:
- a solution to a system of quadratic polynomials,
- a solution to an instance of the Permuted Kernel Problem and
- a witness for a variety of lattice statements (including SIS).

Our sigma protocols have soundness error $1/q'$, where q' is any number bounded by the size of the underlying finite field. This is much better than existing proofs, which have soundness error $2/3$ or $(q'+1)/2q'$. The prover and verifier time of our proofs are $O(q')$. We achieve this by first constructing so-called *sigma protocols with helper*, which are sigma protocols where the prover and the verifier are assisted by a trusted third party, and then eliminating the helper from the proof with a "cut-and-choose" protocol. We apply the Fiat-Shamir transform to obtain signature schemes with security proof in the QROM. We show that the resulting signature schemes, which we call the "MUltivariate quaDratic FIat-SHamir" scheme (MUDFISH) and the "ShUffled Solution to Homogeneous linear SYstem FIat-SHamir" scheme (SUSHSYFISH), are more efficient than existing signatures based on the MQ problem and the Permuted Kernel Problem. Our proof system can be used to improve the efficiency of applications relying on (generalizations of) Stern's protocol. We show that the proof size of our SIS proof is smaller than that of Stern's protocol by an order of magnitude and that our proof is more efficient than existing post-quantum secure SIS proofs.

Keywords: Zero-Knowledge, Post-Quantum digital signatures, SIS, Multivariate cryptography, Permuted Kernel Problem, Silly acronyms

1 Introduction

Zero-knowledge proofs of knowledge and more specifically Sigma protocols are a technique in cryptography that allows a prover to prove to a verifier that they know a value x that satisfies some relation, without revealing any additional information about x [19]. Sigma protocols are useful to build a wide variety of cryptographic applications, including digital signatures, group/ring signatures, e-voting protocols, and privacy-preserving cryptocurrencies. In some cases these sigma protocols are not completely sound, meaning that a cheating prover can convince a verifier he knows some value, without actually knowing it. If a prover can do this with a probability at most ϵ, then ϵ is said to be the *soundness error* of the sigma protocol. The soundness of a sigma protocol can be amplified; by repeating the protocol k times the soundness error of the entire protocol becomes ϵ^k. Therefore, if one repeats a protocol with soundness error ≤ 1 often enough, one can obtain a sound protocol. However, if a large number of repetitions is required, this makes the protocol less efficient and makes applications of the protocol less practical. This is the case for Stern's protocol [34] and the sigma protocols underlying some post-quantum signature schemes [14, 12, 10]. The goal of this paper is to develop new variants of these sigma protocols that have a smaller soundness error, such that fewer repetitions are necessary and such that the overall efficiency of the protocols is improved.

Zero-Knowledge based Post-Quantum signatures. One way to construct a signature scheme is to first construct a zero-knowledge identification scheme and then make it into a non-interactive signature scheme with a transformation such as the Fiat-Shamir transform [17] or the Unruh transform [35]. Looking at the NIST Post-Quantum Standardization project, three of the Round II signature schemes, MQDSS, Picnic, and Dilithium use this approach. MQDSS [13] uses a zero-knowledge proof that, given a multivariate quadratic map $\mathcal{P} : \mathbb{F}_q^n \to \mathbb{F}_q^m$ proves knowledge of a solution $\mathbf{s} \in \mathbb{F}_q^n$ such that $\mathcal{P}(\mathbf{s}) = \mathbf{0}$. Picnic [12] uses an identification scheme constructed using the "MPC-in-the-head" technique [20] that relies on symmetric primitives. Dilithium is a lattice-based signature scheme that relies on the Fiat-Shamir with aborts technique [29]. Another example is PKP-DSS [10], which uses a zero-knowledge proof introduced by Shamir in 1989 for proving knowledge of a solution of an instance of the Permuted Kernel Problem (PKP) [33]. This means that, given a matrix $\mathbf{A} \in \mathbb{F}_q^{m \times n}$ and a vector $\mathbf{v} \in \mathbb{F}_q^n$, the proof system can prove knowledge of a permutation $\pi \in S_n$ such that $\mathbf{A}\mathbf{v}_\pi = 0$, where $\mathbf{v}_\pi$ is the vector obtained by permuting the entries of the vector $\mathbf{v}$ with the permutation π. A drawback of these schemes (with exception of Dilithium) is that the underlying identification schemes have a large soundness error, so a large number of parallel repetitions are required to get a secure signature scheme. This increases the signature sizes and the signing and verification times. For example, the protocol underlying the Picnic signature scheme has a soundness error of $\frac{2}{3}$ and hence requires $k = 219$ repetitions to get the soundness error down to less than 2^{-128}.

Recently, Katz et al. [24] improved on the approach of Picnic by building a zero-knowledge proof from MPC in the preprocessing model, where the parties can use some auxiliary data that was generated during a preprocessing phase. The advantage of moving to the new MPC protocol is that it allows for secure computation with dishonest majority with an arbitrary number of parties n, which results in a zero-knowledge proof with a soundness error of $\frac{1}{n}$. Hence, fewer parallel rounds are required to get a secure signature scheme. A "cut-and-choose" protocol is used to deal with the preprocessing phase, which makes signing and verification slower compared to the original Picnic scheme. This new signature scheme is called Picnic2 and is, together with the original Picnic scheme, one of the Round 2 candidates of the NIST PQC standardization project.

Stern's protocol. In 1993, Stern proposed a code based sigma protocol [34]. For a publicly known parity check matrix $\mathbf{H} \in \mathbb{F}_2^{m \times n}$, syndrome $\mathbf{s} \in \mathbb{F}_2^m$ and weight t, Stern's zero-knowledge proof can prove knowledge of an error vector $\mathbf{e} \in \mathbb{F}_2^n$ with hamming weight t such that $\mathbf{He} = \mathbf{s}$. Internally, Stern's protocol is very similar to Shamir's protocol for PKP, and in fact, Stern's protocol generalizes easily to proving knowledge of a witness of the inhomogeneous PKP (IPKP) relation. The motivation behind Stern's protocol was to obtain a code-based identification scheme (and hence also a signature scheme with the Fiat-Shamir transform). However, Stern's protocol has been used extensively in lattice-based cryptography, because the IPKP relation can be bootstrapped to prove knowledge of a solution to the SIS problem, knowledge of an LWE secret and more complex lattice statements such as proving that a given LWE ciphertext is a valid encryption of a known message satisfying certain constraints [28]. This led to the construction of many advanced primitives from lattices, such as identity-based identification schemes, group signatures (with verifier local revocation), logarithmic size ring signatures and group encryption [28, 25, 27, 26]. Improving Stern's protocol is an important and long-standing open problem because this would improve the efficiency of all these constructions.

Contributions. In this paper we generalize the idea behind Picnic2 [24] to something we call "sigma protocols with helper". Concretely, a sigma protocol with helper is a 3-party protocol between a prover, a verifier and a trusted third party called the "helper". The protocol begins with the helper who honestly generates some auxiliary information that he sends to the verifier. The helper also sends the randomness seed that he used to generate his randomness to the prover. Then, the protocol resumes like a normal sigma protocol. A sigma protocol with helper is similar to a sigma protocol in the Common Reference String (CRS) model, except that the trusted third party sends some secret information (the randomness seed) to the prover and that the trusted third party needs to participate in every execution, rather than just doing the trusted setup once.

We then construct a sigma protocol with helper to prove knowledge of a solution of a system of quadratic equations and a sigma protocol with helper for proving knowledge of a solution of an inhomogeneous PKP instance (i.e. the same relation as the Shamir and Stern protocols). Our proofs have soundness error $\frac{1}{q'}$ and prover time $\Theta(q')$, where q' is any number bounded by the size of the finite fields that are used. This soundness error is much better than existing proofs which have soundness error $\frac{1}{2} + \frac{1}{2q}$ or soundness error $2/3$. We then show how to remove the helper with a "cut-and-choose" protocol, analogous to the approach used by Katz et al. [24]. This transformation gives rise to standard sigma protocols (i.e. without helper) which can then be transformed into signature schemes using the Fiat-Shamir transform or used as a more efficient variant of Stern's protocol as a building block for advanced privacy-preserving constructions.

Note that, even though the soundness error is q', it is not possible to do one-shot proofs if the field size is exponential because the prover time is $\Theta(q')$. However, we can still realize a large practical improvement over existing proofs: The proof size of existing proofs is $\mathcal{O}(\lambda X)$, where λ is the security parameter and X is the proof size of a single iteration of the protocol. In comparison, the proof size of our proofs is $\mathcal{O}(\frac{\lambda}{\log q'}(X + \log q' * |\mathsf{seed}|))$, because the number of iterations is now $O(\frac{\lambda}{\log q'})$, and each iteration incurs an overhead of $\log q' |\mathsf{seed}|$ (a path in a Merkle tree of size q'). In practice, the proof size is often dominated by the $\mathcal{O}(\lambda|\mathsf{seed}|)$ term even for small values of q'. Since X is usually much larger than $|\mathsf{seed}| = \lambda$, this gives a large improvement in practice. X and $|\mathsf{seed}|$ are both linear in λ, so the improvement factor remains the same at higher security levels.

We apply the Fiat-Shamir transform to our Sigma protocol for the MQ relation to get a signature scheme whose security reduces to the problem of finding a solution to a random system of multivariate quadratic polynomials. We call this the "MUltivarite quaDratic FIat-SHamir" scheme (MUDFISH). MUDFISH is more efficient than MQDSS, the existing signature scheme based on the same hard problem. At NIST security level 1, the MUDFISH signatures are roughly half as big as the MQDSS signatures, while our constant-time MUDFISH implementation is roughly twice as fast as the optimized MQDSS implementation that was submitted to the NIST PQC standardization project. Using the Fiat-Shamir transform on our sigma protocol for the PKP relation, we obtain the "ShUffled Solution to Homogeneous linear SYstem FIat-SHamir" scheme (SUSHSYFISH), a signature scheme whose security reduces to finding a solution of a random PKP instance. SUSHSYFISH has smaller signatures than PKP-DSS, the existing scheme based on the PKP problem while being only slightly slower. Moreover, unlike MQDSS and PKP-DSS, the MUDFISH and SUSHSYFISH signature schemes are based on sigma protocols (i.e. 3-round proofs) rather than 5-round proofs, which results in tighter security proofs in the ROM and even allows us to use the recent results of Don et. al. [16] to prove their security in the QROM. A comparison of the signature sizes and signing speed of MUDFISH and multiple instantiations of SUSHSYFISH with those of existing Post-Quantum Fiat-Shamir signatures is given in Fig. 1. Our implementation is available on GitHub [9].

We can improve the lattice-based constructions such as identity-based identification schemes, group signatures (with verifier local revocation), logarithmic size ring signatures and group encryption that rely on Stern's protocol [28, 25, 27, 26], by replacing Sterns protocol by our more efficient proof for IPKP. In particular, we make a case study for the SIS problem, where we see that with our proof system, the proof size is a factor 10 smaller than with Stern's protocol. And smaller than proof sizes arising from other post-quantum exact proofs for SIS, such as using "MPC-in-the-head" techniques [5] or an algebraic approach [11].

Roadmap In sect. 2 we lay out some basic preliminaries required for the remainder of the paper. In Sect. 3 we formalize the notion of a sigma protocol with helper, then we construct sigma protocols with helper for the MQ problem and the Permuted Kernel Problem in sections 4 and 5. In Sect. 6 we show how to convert a sigma protocol with helper in a normal zero-knowledge proof (without helper). Then, we convert our zero-knowledge proofs into signature schemes in Sect. 8, where we also briefly discuss our proof-of-concept implementations. Finally, in Sect. 9 we show how to use the IPKP proof to construct a zero-knowledge proof for the SIS relation, and we compare our SIS proof to existing SIS proofs.

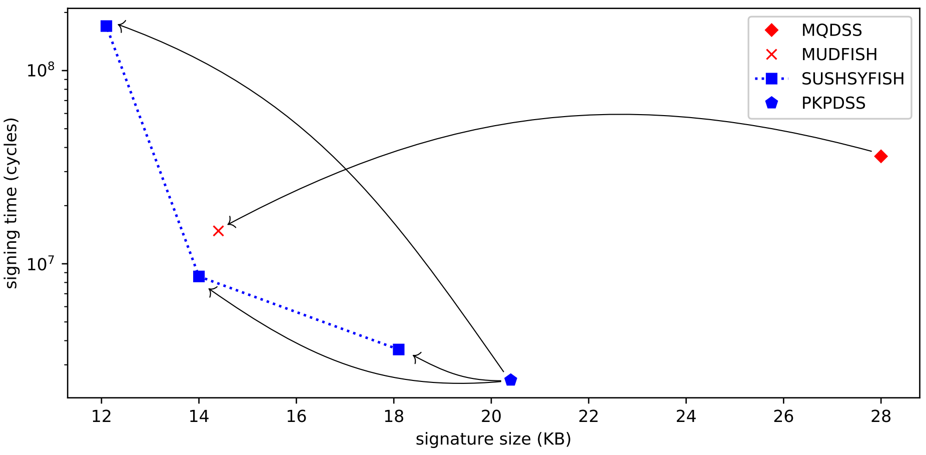


Fig. 1. Comparison of MUDFISH and SUSHSYFISH to existing signatures based on the MQ problem (MQDSS) and PKP problem (PKP-DSS). Cycle counts of picnic and MQDSS are taken from the NIST Round2 submission packages (the optimized, but not AVX2 optimized implementations, updated to take the attack of Kales and Zaverucha into account [23]), cycle counts for PKP-DSS are taken from [10].

2 Preliminaries

2.1 Hard problems

We introduce (variants of) the Permuted Kernel Problem (PKP), the Multivariate quadratic problem (MQ) and the Short Integer Solution problem (SIS), three computationally hard problems that are used in the remainder of the paper.

Permuted Kernel Problem (PKP/IPKP). Given a matrix $\mathbf{A} \in \mathbb{F}_q^{m \times n}$ and a vector $\mathbf{v} \in \mathbb{F}_q^n$ defined over a finite field $\mathbb{F}_q$, the Permuted Kernel Problem is to find a permutation $\pi \in S_n$, such that $\mathbf{A}\mathbf{v}_\pi = 0$, where $\mathbf{v}_\pi$ is the vector obtained by permuting the entries of $\mathbf{v}$ with the permutation π, that is, the vector defined by $(\mathbf{v}_\pi)_i = v_{\pi(i)}$. There is also a inhomogeneous version of the problem, where given $\mathbf{A} \in \mathbb{F}_q^{m \times n}$, $\mathbf{v} \in \mathbb{F}_q^n$ and a target vector $\mathbf{t} \in \mathbb{F}_q^m$, the task is to find a permutation $\pi \in S_n$, such that $\mathbf{A}\mathbf{v}_\pi = \mathbf{t}$.

The permuted kernel problem is a classical NP-Hard problem that was first introduced in cryptography by Shamir, who designed an identification scheme, whose security reduces to the problem of solving a random PKP instance [33]. Several works have introduced new algorithms and time-memory trade-offs for solving the PKP [30, 3, 18, 21], but solving the problem remains prohibitively difficult, even for small parameters (see Table 3).

Subgroup IPKP The Subgroup Inhomogeneous Permuted Kernel Problem (SIPKP) is the same as the IPKP problem, with the additional constraint that the solution is a member of a certain subgroup of S_n. Concretely, a solution to the a SIPKP instance $(\mathbf{A}, \mathbf{v}, \mathbf{t}, H)$, with $\mathbf{A} \in \mathbb{F}_q^{m \times n}, \mathbf{v} \in \mathbb{F}_q^n, \mathbf{t} \in \mathbb{F}_q^m$ and H a subgroup of S_n is a permutation $\pi \in H$ such that $\mathbf{A}\mathbf{v}_\pi = \mathbf{t}$.

Multivariate Quadratic (MQ). Given a multivariate quadratic map $\mathcal{P} : \mathbb{F}_q^n \rightarrow \mathbb{F}_q^m$ of m quadratic polynomials in n variables defined over a finite field $\mathbb{F}_q$, the MQ problem asks to find a solution $\mathbf{s} \in \mathbb{F}_q^n$ such that $\mathcal{P}(\mathbf{s}) = 0$. The best known methods for solving this problem rely on Grobner basis methods or linearization methods in combination with guessing a number of the variables [8, 22]. This is the central problem underlying most of multivariate cryptography, and for random systems $\mathcal{F}$, the hardness of the problem is well understood.

Short Integer Solution (SIS/ISIS). The well known Short Integer Solution problem, introduced in the seminal work of Ajtai [1] asks to, given a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, and a bound β, find a vector $\mathbf{x}$, such that $\mathbf{A}\mathbf{x} = 0$ whose norm is bounded by $||\mathbf{x}|| \leq \beta$. There is also a inhomogenues version of the problem (ISIS), where, given $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, $\mathbf{t} \in \mathbb{Z}_q^n$ and a bound β the taks is to find $\mathbf{x} \in \mathbb{Z}_q^m$ such that $\mathbf{A}\mathbf{x} = \mathbf{t}$, again subject to $||\mathbf{x}|| \leq \beta$. The problem enjoys reductions from worst case lattice problems, and is one of the fundamental problems underlying lattice-based cryptography.

2.2 Commitment schemes

Many sigma protocols, including ours, depend heavily on secure non-interactive commitment schemes. In the remainder of the paper we assume a non-interactive commitment function $\mathsf{Com} : \{0,1\}^\lambda \times \{0,1\}^\star \rightarrow \{0,1\}^{2\lambda}$, that takes as input λ uniformly random bits bits, where λ is the security parameter, and a message $m \in \{0,1\}^\star$ and outputs a 2λ bit long commitment $\mathsf{Com}(\mathsf{bits}, m)$.

Intuitively, the commitment scheme should not reveal anything the message it commits to, and it should not be possible to open the commitment to some different message. These properties are formalized as follows:

Definition 1 (Computational binding.). *For an adversary $\mathcal{A}$ we define its advantage for the commitment binding game as*

$$\mathsf{Adv}^{\mathsf{Binding}}_{\mathsf{Com}}(\mathcal{A}) = \Pr[\mathsf{Com}(\mathsf{bits}, m) = \mathsf{Com}(\mathsf{bits}', m') | (\mathsf{bits}, m, \mathsf{bits}', m') \leftarrow \mathcal{A}(1^\lambda)]$$

We say that Com *is computationally binding if for all polynomial time algorithms $\mathcal{A}$, the advantage $\mathsf{Adv}^{\mathsf{Binding}}_{\mathsf{Com}}(\mathcal{A})$ is a negligible function of the security parameter λ.*

Definition 2 (Computational hiding.). *For an adversary $\mathcal{A}$ we define the advantage for the commitment hiding game for a pair of messages m, m' as*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{Hiding}}_{\mathsf{Com}}(\mathcal{A}, m, m') = \Bigg| &\Pr_{\mathsf{bits} \leftarrow \{0,1\}^\lambda}[1 = \mathcal{A}(\mathsf{Com}(\mathsf{bits}, m)] \\ &- \Pr_{\mathsf{bits} \leftarrow \{0,1\}^\lambda}[1 = \mathcal{A}(\mathsf{Com}(\mathsf{bits}, m')] \Bigg| .\end{aligned}$$

We say that Com *is computationally hiding if for all polynomial time algorithms $\mathcal{A}$, and every pair of messages m, m' the advantage $\mathsf{Adv}^{\mathsf{Hiding}}_{\mathsf{Com}}(\mathcal{A}, m, m')$ is a negligible function of the security parameter λ.*

In our implementations, we use SHAKE256 as commitment function. If we model SHAKE256 as a quantum random oracle, then it satisfies the computational binding and hiding properties.

3 Sigma protocols with helper

This paper introduces two Sigma protocols with helper, which are like normal sigma protocols, with the addition of a trusted third party (called the helper) that runs a setup algorithm based on a random seed at the beginning of each execution of the protocol. The helper then sends some auxiliary information to the verifier and sends the seed value that was used to seed the setup algorithm to the prover. A more formal definition is as follows:

Definition 3 (Sigma protocol with helper). *A protocol is a sigma protocol with helper for relation R with challenge space $\mathcal{C}$ if it is of the form of Fig. 2 and satisfies:*

- **Completeness** *If all parties (Helper, Prover and Verifier) follow the protocol on input $(x, w) \in R$, then the verifier always accepts.*
- **2-Special soundness.** *From an adversary $\mathcal{A}$ that outputs with noticeable probability valid transcripts $(x, \mathsf{aux}, \mathsf{com}, ch, \mathsf{rsp})$ and $(x, \mathsf{aux}, \mathsf{com}, ch', \mathsf{rsp}')$ with $ch \neq ch'$ and where $\mathsf{aux} = Setup(\mathsf{seed})$ for some seed value seed (not necessarily known to the extractor) one can efficiently extract a witness w such that $(x, w) \in R$.*
- **Special honest-verifier zero-knowledge.** *There exists a PPT simulator $\mathcal{S}$ that on input x, a random seed value seed and a random challenge ch outputs a transcript $(x, \mathsf{aux}, \mathsf{com}, ch, \mathsf{rsp})$ with $\mathsf{aux} = Setup(\mathsf{seed})$ that is computationally indistinguishable from the probability distribution of transcripts of honest executions of the protocol on input (x, w) for some w such that $(x, w) \in R$, conditioned on the auxiliary information being equal to aux and the challenge being equal to ch.*

```
Helper($x$)
    $\mathsf{seed} \xleftarrow{\$} \{0,1\}^\lambda$
    $\mathsf{aux} \leftarrow \mathrm{Setup}(\mathsf{seed})$
    Send seed to the prover and aux to the verifier.

        Prover($x, w, \mathsf{seed}$)                                  Verifier($x, \mathsf{aux}$)

    $\mathsf{com}, \mathsf{P\_state} \leftarrow P_1(x, w, \mathsf{seed})$
                                        $\xrightarrow{\mathsf{com}}$
                                                      $ch \xleftarrow{\$} \mathcal{C}$
                                        $\xleftarrow{ch}$
    $\mathsf{rsp} \leftarrow P_2(\mathsf{P\_state}, ch)$
                                        $\xrightarrow{\mathsf{rsp}}$
                                                      return $V(x, \mathsf{aux}, \mathsf{com}, ch, \mathsf{rsp})$
```

Fig. 2. The structure of a sigma protocol with trusted setup.

4 Proving knowledge of a solution to a system of quadratic equations

Two zero-knowledge proofs to prove knowledge of a solution of a system of multivariate quadratic equations over a finite field $\mathbb{F}_q$ were proposed by Sakumoto et al. [32]. The first proof is a 3-round protocol which has soundness error $\frac{2}{3}$, while the second proof is a 5-round protocol with soundness error $\frac{1}{2}+\frac{1}{2q}$, where q is the size of the finite field over which the system of polynomials is defined. The MQDSS [13] signature scheme is obtained by applying the Fiat-Shamir transform to the 5-round protocol of Sakumoto et al. Because the soundness error of $\frac{1}{2}+\frac{1}{2q}$ is rather big, and because the Fiat-Shamir transformation does not tightly preserve security for 5-round protocols [23] a large number (e.g. 184 for the NIST security level I parameter set) of parallel rounds is required to obtain a secure signature scheme.

In this section, we present a sigma protocol with helper to prove knowledge of a solution of a system of multivariate quadratic equations. The scheme improves the knowledge error to only $1/q$, but this comes at the cost of having an honest party that helps the prover and the verifier in their execution of the protocol. Similar to the schemes of Sakumoto et al. the new protocol relies on the fact that if $\mathcal{F} : \mathbb{F}_q^n \rightarrow \mathbb{F}_q^n$ is a multivariate quadratic map of m polynomials in n variables, then the polar form of $\mathcal{F}$, which is defined as

$$\mathcal{G}(\mathbf{x},\mathbf{y}) := \mathcal{F}(\mathbf{x}+\mathbf{y}) - \mathcal{F}(\mathbf{x}) - \mathcal{F}(\mathbf{y}) \tag{1}$$

is linear in both $\mathbf{x}$ and $\mathbf{y}$.

To prove knowledge of a secret $\mathbf{s}$ such that $\mathcal{F}(\mathbf{s}) = \mathbf{v}$ the protocol goes as follows: During the first phase the helper picks a random vector $\mathbf{r_0}$ and commits to linear secret sharings $\mathbf{t}+\mathbf{t}_c = c\mathbf{r}_0, \mathbf{e}+\mathbf{e}_c = c\mathcal{F}(\mathbf{r}_0)$ for each $c \in \mathbb{F}_q$. These commitments are public auxiliary information which the helper sends to the verifier. The helper also sends the seed that he used to generate his randomness to the prover. Then, the prover publishes the masked secret $\mathbf{r}_1 = \mathbf{s} - \mathbf{r}_0$ and commits to the value of $\mathbf{e} + \mathcal{G}(\mathbf{r}_1, \mathbf{t})$. Finally the verifier challenges the prover to reveal $\mathbf{e}_\alpha$ and $\mathbf{t}_\alpha$ for a random choice of $\alpha \in \mathbb{F}_q$ and checks whether the following equation, which is equivalent to Eqn. 1, holds.

$$\mathbf{e} + \mathcal{G}(\mathbf{r}_1, \mathbf{t}) = c\left(\mathcal{F}(\mathbf{s}) - \mathcal{F}(\mathbf{r}_1)\right) - \mathbf{e}_c - \mathcal{G}(\mathbf{r}_1, \mathbf{t}_c)\,, \quad \forall c \in \mathbb{F}_q \tag{2}$$

A more detailed version of the protocol is displayed in Fig. 3.

Theorem 1. *Suppose the used commitment scheme is computationally binding and computationally hiding, then the protocol of Fig. 3 is a sigma protocol with trusted setup as in definition 3 with challenge space $\mathbb{F}_q$.*

Proof. We prove completeness, 2-special soundness and special honest-verifier zero knowledge separately:

Completeness: The fact that in a honest execution of the protocol $\mathbf{x} = \mathbf{e} + \mathcal{G}(\mathbf{r}_1, \mathbf{t})$ follows from Eqn. 2, so completeness follows immediately.

2-Special Soundness: Suppose an extractor is given two transcripts $(\mathsf{aux}, \mathsf{com}, \alpha, (\mathsf{r}, \mathsf{r}_\alpha, \mathbf{r}_1, \mathbf{e}_\alpha, \mathbf{t}_\alpha))$, $(\mathsf{aux}, \mathsf{com}, \alpha, (\mathsf{r}', \mathsf{r}'_\alpha, \mathbf{r}'_1, \mathbf{e}_{\alpha'}, \mathbf{t}_{\alpha'}))$ with $\alpha \neq \alpha'$ that are accepted by the verifier and such that $\mathsf{aux} =$ Setup(seed) (for some seed value unknown to the extractor). Then we show how to extract a witness $\mathbf{s}$ such that $\mathcal{P}(\mathbf{s}) = \mathbf{v}$ if the binding of the commitments does not fail.

Let $\mathbf{x} := \alpha(\mathbf{v} - \mathcal{F}(\mathbf{r}_1)) - \mathbf{e}_\alpha - \mathcal{G}(\mathbf{r}_1, \mathbf{t}_\alpha)$ and $\mathbf{x}' := \alpha'(\mathbf{v} - \mathcal{F}(\mathbf{r}'_1)) - \mathbf{e}_{\alpha'} - \mathcal{G}(\mathbf{r}'_1, \mathbf{t}_{\alpha'})$, then the verifier only accepts if we have $\mathsf{com} = \mathsf{Com}(\mathsf{r}, \mathbf{r}_1, \mathbf{x}) = \mathsf{Com}(\mathsf{r}', \mathbf{r}'_1, \mathbf{x}')$, so the binding property of Com implies that $\mathbf{r}_1 = \mathbf{r}'_1$ and $\mathbf{x} = \mathbf{x}'$.

Even though the extractor does not know $\mathbf{e}, \mathbf{t}, \mathbf{r}_0$ or the commitment random strings $\{\mathsf{r}_c \mid c \in \mathbb{F}_q\}$, the extractor still knows that

$$\mathsf{aux} = \{\mathsf{Com}(\tilde{\mathsf{r}}_c, (c\mathcal{F}(\mathbf{r}_0) - \mathbf{e}, c\mathbf{r}_0 - \mathbf{t})) \mid c \in \mathbb{F}_q\}$$

for *some* values of $\mathbf{e}, \mathbf{t}, \mathbf{r}_0$ and $\{\tilde{\mathsf{r}}_c \mid c \in \mathbb{F}_q\}$, because the helper computed $\mathsf{aux} =$ Setup(seed) honestly.

The verifier only accepts both transcripts if $\mathsf{Com}(\tilde{\mathsf{r}}_\alpha, (\alpha\mathcal{F}(\mathbf{r}_0) - \mathbf{e}, \alpha\mathbf{r}_0 - \mathbf{t})) = \mathsf{Com}(\mathsf{r}_\alpha, (\mathbf{e}_\alpha, \mathbf{t}_\alpha))$ and $\mathsf{Com}(\tilde{\mathsf{r}}_{\alpha'}, (\alpha'\mathcal{F}(\mathbf{r}_0) - \mathbf{e}, \alpha'\mathbf{r}_0 - \mathbf{t})) = \mathsf{Com}(\mathsf{r}'_\alpha, (\mathbf{e}_{\alpha'}, \mathbf{t}_{\alpha'}))$, so the binding property of Com implies that

$$\begin{aligned} \alpha\mathcal{F}(\mathbf{r}_0) - \mathbf{e} &= \mathbf{e}_\alpha\,, & & & \alpha\mathbf{r}_0 - \mathbf{t} &= \mathbf{t}_\alpha\,, \\ \alpha'\mathcal{F}(\mathbf{r}_0) - \mathbf{e} &= \mathbf{e}_{\alpha'} & &\text{and} & \alpha'\mathbf{r}_0 - \mathbf{t} &= \mathbf{t}_{\alpha'}\,. \end{aligned}$$

Substituting this into $\mathbf{x} = \mathbf{x}'$ we get

$$\begin{aligned} &\alpha(\mathbf{v} - \mathcal{F}(\mathbf{r}_1)) + \mathbf{e} - \alpha\mathcal{F}(\mathbf{r}_0) - \mathcal{G}(\mathbf{r}_1, \alpha\mathbf{r}_0 - \mathbf{t}) \\ &= \alpha'(\mathbf{v} - \mathcal{F}(\mathbf{r}_1)) + \mathbf{e} - \alpha'\mathcal{F}(\mathbf{r}_0) - \mathcal{G}(\mathbf{r}_1, \alpha'\mathbf{r}_0 - \mathbf{t})\,, \end{aligned}$$

which simplifies to

$$\begin{aligned} (\alpha - \alpha')\,(\mathcal{F}(\mathbf{r}_1) + \mathcal{F}(\mathbf{r}_0) + \mathcal{G}(\mathbf{r}_0, \mathbf{r}_1) - \mathbf{v}) &= \\ (\alpha - \alpha')\,(\mathcal{F}(\mathbf{r}_0 + \mathbf{r}_1) - \mathbf{v})) &= 0\,, \end{aligned}$$

so $\mathbf{r}_0 + \mathbf{r}_1 = \frac{\mathbf{t}_\alpha - \mathbf{t}_{\alpha'}}{\alpha - \alpha'} + \mathbf{r}_1$ is a solution to $\mathcal{F}(\mathbf{x}) = \mathbf{v}$. Notice that all the values on the right hand side of this equation are included in the 2 transcripts, so extracting the solution from the two transcripts is trivial.

Special honest-verifier zero-knowledge: Define a simulator $\mathcal{S}$, that on input $\mathbf{v}$, a random seed value seed and a random challenge $\alpha \in \mathbb{F}_q$ does the following things:

1. recompute $\mathsf{aux}, \mathsf{r}_\alpha, \mathbf{e}_\alpha$ and $\mathbf{t}_\alpha$ from seed.
2. pick a uniformly random vector $\mathbf{u} \in \mathbb{F}_q^n$.
3. compute $f_{\alpha,\mathbf{e}_\alpha,\mathbf{t}_\alpha}(\mathbf{u})$, where $f_{\alpha,\mathbf{e}_\alpha,\mathbf{t}_\alpha}(\mathbf{x}) := \alpha(\mathbf{v} - \mathcal{F}(\mathbf{x})) - \mathbf{e}_\alpha - \mathcal{G}(\mathbf{x}, \mathbf{t}_\alpha)$.
4. pick commitment randomness r and a commitment com' to $(\mathbf{u}, f_{\alpha,\mathbf{e}_\alpha,\mathbf{t}_\alpha}(\mathbf{u}))$.
5. output $(\mathsf{aux}, \mathsf{com}', \alpha, (\mathsf{r}, \mathsf{r}_\alpha, \mathbf{u}, \mathbf{e}_\alpha, \mathbf{t}_\alpha))$.

Then the Simulator is identical to an honest prover, except for step 2, where the honest prover sets $\mathbf{u}$ equal to $\mathbf{s} - \mathbf{r}_0$ rather than a uniformly random value. It is clear that $(\alpha, \mathsf{r}, \mathsf{r}_\alpha, \mathbf{u}, \mathbf{e}_\alpha, \mathbf{t}_\alpha)$ and $(\alpha, \mathsf{r}, \mathsf{r}_\alpha, \mathbf{s} - \mathbf{r}_0, \mathbf{e}_\alpha, \mathbf{t}_\alpha)$ are both uniformly distributed in $\mathbb{F}_q \times \{0,1\}^{2\lambda} \times \left(\mathbb{F}_q^n\right)^3$ and hence follow the same distribution. Since com and com_α are completely determined by $(\alpha, \mathsf{r}, \mathsf{r}_\alpha, \mathbf{s} - \mathbf{r}_0, \mathbf{e}_\alpha, \mathbf{t}_\alpha)$ it follows that $(\mathsf{com}_\alpha, \mathsf{com}', \alpha, \mathsf{r}, \mathsf{r}_\alpha, \mathbf{u}, \mathbf{e}_\alpha, \mathbf{t}_\alpha)$ and $(\mathsf{com}_\alpha, \mathsf{com}, \alpha, \mathsf{r}, \mathsf{r}_\alpha, \mathbf{s} - \mathbf{r}_0, \mathbf{e}_\alpha, \mathbf{t}_\alpha)$ also follow the same distribution. Finally, since the commitments $\mathsf{com}_{c \neq \alpha}$ are never opened, it follows from the hiding property of the commitment scheme with the standard hybrid argument that $(\mathsf{aux}, \mathsf{com}', \alpha, \mathsf{r}, \mathsf{r}_\alpha, \mathbf{u}, \mathbf{e}_\alpha, \mathbf{t}_\alpha)$ is computationally indistinguishable from $(\mathsf{aux}, \mathsf{com}, \alpha, \mathsf{r}, \mathsf{r}_\alpha, \mathbf{s} - \mathbf{r}_0, \mathbf{e}_\alpha, \mathbf{t}_\alpha)$.

5 Proving knowledge of a solution to a (inhomogeneous) PKP instance

In this section we give a Sigma protocol with helper with challenge space $\mathbb{F}_p$ to prove knowledge of a solution for an inhomogeneous PKP instance, i.e. given $\mathbf{A}, \mathbf{v}, \mathbf{t}$ our proof system proves knowledge of a permutation π such that $\mathbf{A}\mathbf{v}_\pi = \mathbf{t}$. The soundness error of our proof is only $1/p$, which is much better than the 5-round proof of Shamir, which has a soundness error of $\frac{1}{2} + \frac{1}{2p}$ [33], and Stern's 3-round protocol, which has a soundness error of $2/3$ [34].

To prove knowledge of a solution π to the instance $(\mathbf{A}, \mathbf{v}, \mathbf{t})$ the protocol goes as follows: The helper picks a random vector $\mathbf{r} \in \mathbb{F}_p^n$, and a random permutation $\sigma \in S_n$, it then commits to $\mathbf{r} + c\mathbf{v}_\sigma$ for all values of $c \in \mathbb{F}_p$. The helper sends

```
Helper($\mathcal{F}$)
    $\mathsf{seed} \xleftarrow{\$} \{0,1\}^\lambda$
    Generate $\mathbf{e} \in \mathbb{F}_q^m$ and $\mathbf{t}, \mathbf{r}_0 \in \mathbb{F}_q^n$ from $\mathsf{seed}$.
    for each $c$ in $\mathbb{F}_q$ do
        $\mathbf{e}_c \leftarrow c\mathcal{F}(\mathbf{r_0}) - \mathbf{e}$
        $\mathbf{t}_c \leftarrow c\mathbf{r_0} - \mathbf{t}$
        Generate commitment randomness $\mathsf{r}_c \in \{0,1\}^\lambda$ from $\mathsf{seed}$.
        $\mathsf{com}_c \leftarrow \mathsf{Com}(\mathsf{r}_c, (\mathbf{e}_c, \mathbf{t}_c))$
    end for
    $\mathsf{aux} \leftarrow [\mathsf{com}_c |$ for $c \in \mathbb{F}_q]$
    Send $\mathsf{seed}$ to the prover and $\mathsf{aux}$ to the verifier.

    Prover($\mathcal{F}, \mathbf{s}, \mathsf{seed}$)                          Verifier($\mathcal{F}, \mathbf{v}, \mathsf{aux}$)

    Regenerate $\mathbf{e}, \mathbf{t}, \mathbf{r}_0$ from $\mathsf{seed}$.
    $\mathbf{r}_1 \leftarrow \mathbf{s} - \mathbf{r}_0$
    $\mathsf{r} \leftarrow \{0,1\}^\lambda$
    $\mathsf{com} \leftarrow \mathsf{Com}(\mathsf{r}, (\mathbf{r}_1, \mathbf{e} + \mathcal{G}(\mathbf{r}_1, \mathbf{t})))$

                                  $\xrightarrow{\mathsf{com}}$
                                                                $\alpha \xleftarrow{\$} \mathbb{F}_q$
                                  $\xleftarrow{\alpha}$
    Recompute $\mathsf{r}_\alpha, \mathbf{e}_\alpha, \mathbf{t}_\alpha$ from $\mathsf{seed}$.
                                  $\xrightarrow{(\mathsf{r}, \mathsf{r}_\alpha, \mathbf{r}_1, \mathbf{e}_\alpha, \mathbf{t}_\alpha)}$
                                                                $\mathbf{x} \leftarrow \alpha(\mathbf{v} - \mathcal{F}(\mathbf{r}_1)) - \mathbf{e}_\alpha - \mathcal{G}(\mathbf{r}_1, \mathbf{t}_\alpha)$
                                                                $b_1 \leftarrow \mathsf{com} = \mathsf{Com}(\mathsf{r}, (\mathbf{r}_1, \mathbf{x}))$
                                                                $b_2 \leftarrow \mathsf{com}_\alpha = \mathsf{Com}(\mathsf{r}, (\mathbf{e}_\alpha, \mathbf{t}_\alpha))$
                                                                return $b_1 \wedge b_2$
```

Fig. 3. A sigma protocol with helper for proving knowledge of a solution to the MQ problem.

these commitments as public auxiliary information to the verifier, and he sends the seed that he used to generate his randomness to the prover. Then the prover sends $\rho = \pi\sigma^{-1}$ to the verifier and commits to the value of $\mathbf{Ar}_{\pi\sigma^{-1}}$. Finally, the verifier challenges the prover to reveal $\mathbf{x} = \mathbf{r} + \alpha\mathbf{v}_\sigma$ for a random choice of α. Once the prover reveals $\mathbf{x}$ the verifier checks if $\mathbf{Ax}_\rho - \alpha\mathbf{t} = \mathbf{A}\left(\mathbf{r}_{\pi\sigma^{-1}} + \alpha\mathbf{v}_\pi\right) - \alpha\mathbf{t} = \mathbf{Ar}_{\pi\sigma^{-1}}$. For a more detailed description of the protocol we refer to Fig. 4.

Theorem 2. *Suppose the used commitment scheme is computationally binding and computationally hiding, then the protocol of Fig. 4 is a sigma protocol with trusted setup as in definition 3 with challenge space $\mathbb{F}_p$.*

Proof. We prove completeness, 2-special soundness and special honest-verifier zero knowledge separately:

Completeness: In an honest execution of the protocol we have

$$\mathbf{y} = \mathbf{Ax}_\rho - \alpha\mathbf{t} = \mathbf{A}\left(\mathbf{r}_{\pi\sigma^{-1}} + \alpha\mathbf{v}_\pi\right) - \alpha\mathbf{t}\,,$$

so if π is a solution to the PKP instance $(\mathbf{A}, \mathbf{v}, \mathbf{t})$, then $\mathbf{Av}_\pi = \mathbf{t}$, which means $\mathbf{y} = \mathbf{Ar}_{\pi\sigma^{-1}}$ and hence the completeness follows from the completeness of the commitment scheme.

2-Special Soundness: Given two transcripts $(\mathsf{aux}, \mathsf{com}, \alpha, (\mathsf{r}, \mathsf{r}_\alpha, \rho, \mathbf{x}))$ and $(\mathsf{aux}, \mathsf{com}, \alpha', (\mathsf{r}', \mathsf{r}'_\alpha, \rho', \mathbf{x}'))$ with $\alpha \neq \alpha'$ that are accepted by the verifier and such that aux =Setup(seed), for some value of seed (not necessarily known to the extractor). Then, if the binding of the commitment scheme does not fail (which, by assumption, happens with overwhelming probability), one can efficiently extract a witness π such that $\mathbf{Av}_\pi = \mathbf{t}$.

Let $\mathbf{y} := \mathbf{Ax}_\rho - \alpha\mathbf{t}$ and $\mathbf{y}' := \mathbf{Ax}'_{\rho'} - \alpha'\mathbf{t}$, then the verifier only accepts if we have $\mathsf{com} = \mathsf{Com}(\mathsf{r}, (\rho, \mathbf{y})) = \mathsf{Com}(\mathsf{r}', (\rho', \mathbf{y}'))$, so the binding property of Com implies that $\rho = \rho'$ and $\mathbf{y} = \mathbf{y}'$.

Note that even though the extractor does not know $\mathbf{r}, \sigma$ or any of the commitment randomness strings r_c, he still knows that aux is of the form

$$\mathsf{aux} = \{\mathsf{Com}(\mathsf{r_c}, \mathbf{r} + c\mathbf{v}_\sigma) \mid c \in \mathbb{F}_q\}$$

for *some* values of $\mathbf{r}, \sigma$ and $\{\mathsf{r_c}\}_{c\in\mathbb{F}_q}$, because we can assume the helper computed aux =Setup(seed) honestly.

The verifier only accepts both transcripts if $\mathsf{Com}(\mathsf{r}_\alpha, \mathbf{r} + \alpha\mathbf{v}_\sigma) = \mathsf{Com}(\mathsf{r}_\alpha, \mathbf{x})$ and $\mathsf{Com}(\mathsf{r}_{\alpha'}, \mathbf{r} + \alpha'\mathbf{v}_\sigma) = \mathsf{Com}(\mathsf{r}_{\alpha'}, \mathbf{x}')$, so the binding property of Com implies that $\mathbf{x} = \mathbf{r} + \alpha\mathbf{v}_\sigma$, and that $\mathbf{x}' = \mathbf{r} + \alpha'\mathbf{v}_\sigma$.

Putting everything together we get

$$\mathbf{A}\left(\mathbf{r}_\rho + \alpha \mathbf{v}_{\rho\sigma}\right) - \alpha \mathbf{t} = \mathbf{A}\left(\mathbf{r}_\rho + \alpha' \mathbf{v}_{\rho\sigma}\right) - \alpha' \mathbf{t}$$

which simplifies to

$$(\alpha - \alpha')\left(\mathbf{A}\mathbf{v}_{\rho\sigma} - \mathbf{t}\right) = 0,$$

so $\rho\sigma$ is a solution to the instance of the permuted kernel problem. The value of ρ is known to the extractor because it is included in the transcripts, and the value of σ can be deduced from $\alpha, \alpha', \mathbf{x}, \mathbf{x}'$ and $\mathbf{v}$, because $\mathbf{x} - \mathbf{x}' = (\alpha - \alpha')\mathbf{v}_\sigma$. (If the entries of $\mathbf{v}$ are not unique, multiple values of σ are possible, but they will all give valid solutions to the PKP problem.)

Special honest-verifier zero knowledge: Define a simulator $\mathcal{S}$, that on input $\mathbf{A}, \mathbf{v}$, a random seed value seed and a random challenge $\alpha \in \mathbb{F}_p$ does the following things:

1. recompute $\mathsf{aux}, \mathsf{r}_\alpha$ and $\mathbf{x} = \mathbf{r} + \alpha \mathbf{v}_\sigma$ from seed.
2. pick a uniformly random permutation $\tau \in S_n$.
3. produce commitment randomness r, and a commitment com' to $(\tau, \mathbf{A}\mathbf{x}_\tau)$.
4. output $(\mathsf{aux}, \mathsf{com}', \alpha, (\mathsf{r}, \mathsf{r}_\alpha, \tau, \mathbf{A}\mathbf{x}_\tau))$.

Then the Simulator is identical to an honest prover, except for step 2, where the honest prover sets ρ equal to $\pi\sigma^{-1}$ rather than a uniformly random value. It is clear that $(\alpha, \mathsf{r}, \mathsf{r}_\alpha, \tau, \mathbf{A}\mathbf{x}_\tau)$ and $(\alpha, \mathsf{r}, \mathsf{r}_\alpha, \rho, \mathbf{A}\mathbf{x}_\rho)$ are both uniformly distributed in $\mathbb{F}_q \times \{0,1\}^{2\lambda} \times S_n \times \mathbb{F}_q^n$ and hence follow the same distribution. Since com and com_α are completely determined by $(\alpha, \mathsf{r}, \mathsf{r}_\alpha, \rho, \mathbf{A}\mathbf{x}_\rho)$ it follows that $(\mathsf{com}_\alpha, \mathsf{com}', \alpha, \mathsf{r}, \mathsf{r}_\alpha, \tau, \mathbf{A}\mathbf{x}_\tau)$ and $(\mathsf{com}_\alpha, \mathsf{com}, \alpha, \mathsf{r}, \mathsf{r}_\alpha, \rho, \mathbf{A}\mathbf{x}_\rho)$ also follow the same distribution. Finally, since the commitments $\mathsf{com}_{c \neq \alpha}$ are never opened, it follows from the hiding property of the commitment scheme and the standard hybrid argument that $(\mathsf{aux}, \mathsf{com}', \alpha, (\mathsf{r}, \mathsf{r}_\alpha, \tau, \mathbf{A}\mathbf{x}_\tau))$ is computationally indistinguishable from $(\mathsf{aux}, \mathsf{com}, \alpha, (\mathsf{r}, \mathsf{r}_\alpha, \rho, \mathbf{A}\mathbf{x}_\rho))$.

6 Removing the helper

In this section, we show how to transform a Sigma protocol with helper into a standard zero-knowledge proof of knowledge (without helper). We use the same “Cut-and-choose” approach that was used by Katz et al. to get rid of the preprocessing phase [24].

```
Helper(v)
    $\mathsf{seed} \xleftarrow{\$} \{0,1\}^\lambda$
    Generate $\mathbf{r} \in \mathbb{F}_p^n$ and $\sigma \in S_n$ from $\mathsf{seed}$.
    for each $c$ in $\mathbb{F}_p$ do
        Generate commitment randomness $\mathsf{r}_c \in \{0,1\}^\lambda$ from $\mathsf{seed}$.
        $\mathsf{com}_c \leftarrow \mathsf{Com}(\mathsf{r}_c, \mathbf{r} + c\mathbf{v}_\sigma)$
    end for
    $\mathsf{aux} \leftarrow [\mathsf{com}_c|$ for $c \in \mathbb{F}_p]$
    Send $\mathsf{seed}$ to the prover and $\mathsf{aux}$ to the verifier.

        Prover($\mathbf{A}, \mathbf{v}, \pi, \mathsf{seed}$)                          Verifier($\mathbf{A}, \mathbf{v}, \mathbf{t}, \mathsf{aux}$)

    Regenerate $\mathbf{r}, \sigma$ from $\mathsf{seed}$.
    $\rho \leftarrow \pi\sigma^{-1}$
    $\mathsf{r} \leftarrow \{0,1\}^\lambda$
    $\mathsf{com} \leftarrow \mathsf{Com}(\mathsf{r}, (\rho, \mathbf{A}\mathbf{r}_\rho))$

                                        $\xrightarrow{\mathsf{com}}$
                                                      $\alpha \xleftarrow{\$} \mathbb{F}_q$

                                        $\xleftarrow{\alpha}$
    Recompute $\mathsf{r}_\alpha$ and
    $\mathbf{x} = \mathbf{r} + \alpha\mathbf{v}_\sigma$ from $\mathsf{seed}$.

                                        $\xrightarrow{(\mathsf{r}, \mathsf{r}_\alpha \rho, \mathbf{x})}$
                                                      $\mathbf{y} \leftarrow \mathbf{A}\mathbf{x}_\rho - \alpha\mathbf{t}$
                                                      $b_1 \leftarrow \mathsf{com} = \mathsf{Com}(\mathsf{r}, (\rho, \mathbf{y}))$
                                                      $b_2 \leftarrow \mathsf{com}_\alpha = \mathsf{Com}(\mathsf{r}_\alpha, \mathbf{x})$
                                                      return $b_1 \wedge b_2$
```

Fig. 4. A sigma protocol with helper for proving knowledge of a solution to the inhomogeneous PKP problem.

The idea is to let the prover pick k seeds $\mathsf{seed}_1, \cdots, \mathsf{seed}_k$ and generate k sets of auxiliary information $\mathsf{aux}_i = \text{Setup}\ (\mathsf{seed}_i)$ which the prover sends to the verifier, along with the first messages of the protocol $\mathsf{com}_i = P_1(x, w, \mathsf{seed}_i)$ for all i from 1 to k. The verifier then picks a random index I and a single challenge $ch \in \mathcal{C}$ and sends this to the prover. The prover then sends seed_i for $i \neq I$ as well as a response rsp to the challenge at index I. Using the seeds, the verifier then checks if all the auxiliary information $\mathsf{aux}_{i \neq I}$ was generated properly and checks if rsp is a correct response to the challenge at index I. The details of the protocol are displayed in Fig. 5. We prove that this is a honest-verifier zero knowledge protocol with soundness error $\max(\frac{1}{k}, \frac{1}{|\mathcal{C}|})$.

Theorem 3. *Let (Setup, P_1, P_2, V) be a sigma protocol with helper and challenge space $\mathcal{C}$, if the used commitment scheme is hiding, then the protocol of Fig. 5 is an honest-verifier zero knowledge proof of knowledge with challenge space* $\{1, \cdots, k\} \times \mathcal{C}$ *and* $\max(k, |\mathcal{C}|) + 1$*-special soundness (and hence it has soundness error* $\max(\frac{1}{k}, \frac{1}{|\mathcal{C}|})$*).*

Proof. We prove completeness, special soundness and special honest-verifier zero knowledge separately.

Completeness: Follows immediately from the completeness of the underlying Sigma protocol with trusted setup.

($\max(k, |\mathcal{C}|) + 1$)-special soundness: If there are $\max(k, |\mathcal{C}|) + 1$ valid transcripts then there are at least two valid transcripts with different values of I, which implies that all k setups were done honestly. The pigeon hole principle says there are at least two accepted transcripts with the same value of I, but different ch, so the extractor can use special soundness of the underlying Sigma protocol with trusted setup to extract a witness w.

Special Honest-verifier zero-knowledge: On input (I, ch), the simulator generates all the setups honestly, and commits to random dummy values to create the commitments $\mathsf{com}_{i \neq I}$. The simulator then uses the simulator of the underlying sigma protocol with trusted setup to simulate the transcript at index I. Indistinguishability follows from the hiding property of the commitment scheme and the honest-verifier zero-knowledge property of the underlying sigma protocol with trusted setup.

```
**Prover**                                                          **Verifier**

for $i \in \{1, \cdots, k\}$ do
    $\mathsf{seed}_i \xleftarrow{\$} \{0,1\}^\lambda$
    $\mathsf{aux}_i \leftarrow \mathrm{Setup}(\mathsf{seed}_i)$
    $\mathsf{com}_i \leftarrow P_1(x, w, \mathsf{seed}_i)$
end for
                              $\xrightarrow{\mathsf{aux}_i, \mathsf{com}_i \forall i}$
                                                                    $I \xleftarrow{\$} \{1, \cdots, k\}$
                                                                    $ch \xleftarrow{\$} \mathcal{C}$
                              $\xleftarrow{(I, ch)}$
$\mathsf{rsp} \leftarrow P_2(x, w, \mathsf{seed}_I, \mathsf{com}, ch)$
                              $\xrightarrow{\mathsf{seed}_i \forall i \neq I, \mathsf{rsp}}$
                                                                    if $\exists i \neq I : \mathsf{aux}_i \neq \mathrm{Setup}(\mathsf{seed}_i)$
                                                                    then
                                                                        return $0$
                                                                    end if
                                                                    return $V(x, \mathsf{aux}, \mathsf{com}, ch, \mathsf{rsp})$
```

Fig. 5. A zero knowledge proof (without trusted setup) from a Sigma protocol with trusted setup.

7 Optimizations

In this section, we describe optimizations for the MQ and PKP zero-knowledge proofs with trusted setup, as well as for the transformation that removes the trusted setup. The first two optimizations are applications of standard techniques and the last optimization was proposed by Katz et al. [24], and proven secure by Baum and Nof [5].

Hashing and Merkle trees. In both the MQ proof and the PKP proof the auxiliary information consists of q commitments com_i for $i \in \mathbb{F}_q$, but only one of these commitments, com_α, is opened in each honest execution of the protocol. To reduce the communication cost (and hence the signature size after the Fiat-Shamir transform) we can build a Merkle tree on these commitments and only send the root of the tree. Then the prover includes in his response the $\lceil \log_2(q) \rceil$ nodes of the Merkle tree required to reconstruct the root of the Merkle tree.

When we are doing the transformation to get rid of the trusted party, we do not have to send all the k roots separately. Instead, it suffices to send a hash of all the roots to the verifier. Then, during verification, the verifier recomputes all the roots (either from seed_i if $i \neq I$, or through the verification algorithm if $i = I$) and hashes the roots to verify that they were correct.

The prover sends k commitments com_i, but only the commitment com_I is used. Therefore, similar to the first optimization, the prover can build a Merkle tree on his commitments and send the root to the verifier. Then, he includes com_I and some nodes of the Merkle tree in his response, so the verifier can recompute the root and authenticate com_I.

Sending fewer seeds. The prover chooses k seed values and sends all but one of these seeds to the verifier. We can use a tree strategy to reduce the communication cost. The prover constructs a binary tree of seed values. First, he picks the value of the root at random. Then, the value of each internal node is used to seed a PRNG which generates the values of its two children. In the end, the leaf nodes act as the seed_i values. Now, instead of sending $k-1$ seed values, the prover can send $\lceil \log_2(k) \rceil$ node values in the tree and the prover can recompute the $k-1$ seeds himself (but not seed_I).

Smaller challenge space. For some applications, the finite field $\mathbb{F}_q$ is so large that it would not be practical to compute Merkle trees of size q. In that case, we can simply reduce the challenge space to some subset of $\mathbb{F}_q$ of size $q' \leq q$. The soundness error of the scheme then becomes $1/q'$ instead of $1/q$.

Beating parallel repetition. The basic scheme has soundness error $\frac{1}{q'}$, so to reach a soundness error of $2^{-\lambda}$ we would need to perform $r = \left\lceil \frac{\lambda}{log_2(q')} \right\rceil$ parallel executions of the protocol. The optimization of Katz et al. [24] gives a more efficient approach: The idea is that instead of letting the verifier choose 1 out of k setups to execute, we now let him choose τ out of M setups to execute. Now suppose a cheating prover does $e \leq \tau$ out of the M setups incorrectly. Since he cannot produce seed_i values for the cheated setups, he can only convince the verifier if all the setups in which he cheated end up being executed. This happens with probability $\binom{M-e}{\tau-e} \cdot \binom{M}{\tau}^{-1}$. Then, the prover still needs to generate responses for $\tau - e$ honest setups, which he can do with probability at most $\left(\frac{1}{q'}\right)^{\tau-e}$. Therefore the soundness error of the adapted scheme is bounded by

$$\max_{0 \leq e \leq \tau} \frac{\binom{M-e}{\tau-e}}{\binom{M}{\tau} q'^{\tau-e}}.$$

For a more formal proof we refer to [5].

Example 1. Suppose $q = 128$, then without the optimization, we would need 19 parallel executions of the basic protocol to reach a soundness error of 2^{-128}, which amounts to $19 * 128 = 2432$ setups and 19 executions of the protocol. With the optimization, it turns out that 916 setups and 20 executions are sufficient. So, in this case, the optimization reduces the number of setups by a factor 2.6 at the cost of a single extra execution.

8 Signature schemes

In this section, we apply the Fiat-Shamir transformation to the zero-knowledge proofs for MQ and PKP (after applying the transformation of Sect. 6) to obtain 2 signature schemes. We call these schemes the "MUltivariate quaDratic FIat-SHamir" scheme (MUDFISH) and the "ShUffled Solution to Homogeneous linear SYstem FIat-SHamir" scheme (SUSHSYFISH). First, we observe that the recent results on Post-Quantum Fiat-Shamir by Don et al. [15] apply and thus that our signature scheme are provably secure in the QROM (with non-tight reductions). We then give some generic optimizations for the signature scheme and parameter choices for MUDFISH and SUSHSYFISH. We provide a proof-of-concept implementation to show that MUDFISH and SUSHSYFISH are more efficient than existing signature schemes based on the MQ and PKP assumptions (i.e. MQDSS and PKP-DSS respectively) in terms of signature size and speed (on the NIST reference platform).

8.1 Fiat-Shamir transform

The Fiat-Shamir transform allows us to convert the sigma protocols for MQ and PKP into signatures. The idea is that instead of letting the verifier choose the challenge at random, we derive the challenge deterministically from the commitment and the message that we want to sign. Concretely, to sign a message m, the signer executes the first part of the identification scheme to produce a commitment com, then he derives a challenge ch by applying a random oracle to $\mathsf{com}|m$. Finally, the signer completes his part of the identification scheme to produce the response rsp. The signature is then simply $(\mathsf{com}, \mathsf{resp})$. To verify a signature $(\mathsf{com}, \mathsf{resp})$ for a message m, the verifier simply computes ch by querying the random oracle at $\mathsf{com}|m$, and then he accepts the signature if and only if $(\mathsf{com}, ch, \mathsf{resp})$ is a valid transcript of the identification protocol. Using the results of [15], it is straightforward to prove that MUDFISH and SUSHSYFISH are strongly unforgeable in the QROM.

Theorem 4. *Assume that a hash function modeled as a Quantum Random Oracle is used as commitment scheme and that a Quantum random oracle model is used as PRG, then the non-optimized variants of MUDFISH and SUSHSYFISH signature schemes are strongly existentially unforgeable in the QROM.*

Proof. (The argument is similar to the proof for the FS variant of Picnic, see Sect. 6.1 of [15].) First, we prove that the Setup function is collapsing: If we model the commitment functions as Quantum random oracles, then they are collapsing [36]. In both the MUDFISH and SUSHYFISH schemes, the Setup algorithm consists of expanding a randomness seed, computing some values based on the output of the PRG, and committing to them. In both cases, the PRG output is more than three times longer than the input, so this function is injective with overwhelming probability. Also, it is easily verified that the computing of the values from the output of the PRG is injective. Since the concurrent composition of collapsing functions is collapsing [16] and composing a collapsing function with an injective function preserves collapsingness, it follows that the entire Setup algorithm is collapsing.

Since the responses of the sigma protocol only consist of openings of commitments (which are preimages to Com), and preimages to the Setup function it follows from the collapsingness of Com and Setup that the MUDFISH and SUSHSYFISH sigma protocols have quantum computationally unique responses. Moreover, the protocols have k-special soundness, so theorem 25 of [15] says that the non-optimized versions of MUDFISH and SUSHSYFISH are quantum computational proofs of knowledge. Together with their theorem 22, this implies that MUDFISH and SUSHSYFISH are sEUF-CMA secure.

8.2 MUDFISH

Parameter choices For ease of implementation, we have chosen to use the same finite field $\mathbb{F}_4$ for all the parameter sets. To have a fair comparison with the MQDSS scheme, and to avoid the technicalities of choosing secure parameters for the MQ problem, we use the parameters proposed in the MQDSS submission to the NIST PQC standardization project. These parameter choices for the MQ problem are displayed in Table 1.

We still need to pick parameters for the ZK proof (i.e. τ, the number of executions and M, the number of setups). The choice of τ and M allows for a trade-off: If one is willing to increase τ, which mainly impacts signature size, then one can decrease M, which mainly impacts signing and verification time.

NIST PQC Security Level	q	$n = m$	Best classical attack gates	Best quantum attack gates	depth
I	4	88	2^{156}	2^{93}	2^{83}
III	4	128	2^{230}	2^{129}	2^{119}
V	4	160	2^{290}	2^{158}	2^{147}

Table 1. parameters for the MQ problem used by MUDFISH, and the complexity of solving them with the Crossbred algorithm. The parameter sets and the complexity estimates are taken from Table 8.4 of [14].

NIST PQC Security Level	Parameters q	n	M	τ	$\lvert\mathsf{pk}\rvert$ (B)	$\lvert\mathsf{sk}\rvert$ (B)	$\lvert\mathsf{sig}\rvert$ (KB)	KeyGen (Mc)	Sign (Mc)	Verify (Mc)
I	4	88	191	68	38	16	14.4	2.3	14.8	15.3
III	4	128	256	111	56	24	32.9	7.2	51.3	49.6
V	4	160	380	136	72	32	55.6	14.2	140.4	139.3

Table 2. parameters for MUDFISH, key and signature sizes and performance measurements (average over 1000 signatures).

Implementation results The signing and verification algorithms require to do a lot of setups and executions of the ZK proof on independent data. We take advantage of this by fitting data from 64 independent rounds into one word. Hence, we can do 64 setups or 64 executions of the protocol in parallel on a 64-bit machine. Since the MUDFISH algorithm is inherently constant-time, there was no performance penalty for making the implementation constant-time. Our proof-of-concept implementation uses SHAKE256 as hash function and to expand randomness. The performance results of the implementation are displayed in Table 2. We see that MUDFISH is more efficient than MQDSS: Comparing the parameter sets that achieve NIST security level I, the signatures of MUDFISH are only half as big as those of MQDSS. At the same time, the signing and verification speed of our proof-of-concept implementation of MUDFISH is a factor 2.5 and 1.8 faster than those of the optimized implementation of MQDSS submitted to the second round of the NIST PQC standardization project. We leave the task of making an AVX2 optimized implementation of MUDFISH and comparing its performance to the AVX2 optimized implementation of MQDSS for future work.

8.3 SUSHSYFISH

Parameter choices An advantage of building cryptography on PKP is that the best attack algorithms are quite simple and easy to analyze. We use the PKP parameter sets proposed by Faugère et al. [10] to achieve the NIST security levels 1, 3 and 5. The choice of the remaining parameters q', τ and M allows for a trade-off between signature size and signing and verification speed. For each of the NIST PQC security levels 1, 3 and 5 we propose a parameter set which aims to be fast ($q' = 4$), a parameter sets which aims to have small signatures $q' = 128$ and an intermediate parameter set $q' = 16$.

NIST PQC Security level		q	n	m	q'	M	τ	|pk| (B)	|sk| (B)	|sig| (KB)	KeyGen (Mc)	Sign (Mc)	Verify (Mc)
	Fast	997	61	28	4	191	68	72	16	18.1	0.1	3.6	1.7
I	Middle	997	61	28	16	250	36	72	16	14.0	0.1	8.6	6.0
	Compact	997	61	28	128	916	20	72	16	12.1	0.1	170	169
	Fast	1409	87	42	4	256	111	108	24	43.7	0.1	7.3	3.3
III	Middle	1409	87	42	16	452	51	108	24	30.8	0.1	22.7	16.5
	Compact	1409	87	42	128	1357	30	108	24	27.1	0.1	365	342
	Fast	1889	111	55	4	380	136	142	32	72.8	0.2	12.1	5.8
V	Middle	1889	111	55	16	643	67	142	32	54.9	0.2	25.7	18.0
	Compact	1889	111	55	128	2096	39	142	32	47.5	0.2	602	567

Table 3. parameters for SUSHSYFISH, key and signature sizes and performance measurements (average over 1000 signatures).

Making the implementation constant-time. Most of the SUSHSYFISH algorithm is inherently constant-time, except for some operations involving permutations such as composing permutations, applying a permutation to a vector and sampling random permutations. Naive implementations of these operations involve accessing memory at secret indices, which could make the implementation vulnerable to cache timing attacks. In our implementation, we leverage the *djbsort* constant-time sorting software library [7] to do these operations in constant-time. For example, to apply a permutation $\pi \in S_n$ to a vector $\mathbf{v} \in \mathbb{F}_p^n$ we first construct a list of integers x_i, such that the high-order bits of x_i correspond to π_i, and such that the low-order bits of x_i correspond to v_i. We then call the *djbsort* library to sort this list of integers in constant-time, and we extract the low-order bits from the sorted list, which correspond to $\mathbf{v}_\pi$. Since the performance bottleneck of SUSHSYFISH is hashing, a slight increase in the cost of the operations involving permutations has a negligible effect on the total performance of the signature scheme.

Implementation results. Our implementation uses SHAKE256 as hash function and to expand randomness. The signing and verification times are dominated by the use of SHAKE256, and hence there is a lot of potential for speedups by choosing different symmetric primitives or by parallelizing independent calls of the SHAKE function. The key and signature sizes and the performance measurements for the 9 proposed parameter sets are displayed in Table 3. We see that SUSHSYFISH has smaller signatures than PKP-DSS while being only slightly slower. For NIST PQC security level I, the performance of the "Fast" SUSHSYFISH parameter set is the close to the performance of

PKP-DSS: Signatures are 12% smaller, while with the current implementations signing and verification are 44% and 80% slower respectively. The "Middle" and "Fast" parameter sets offer more compact signatures at the cost of slower signing and verification.

Comparison to previous works. In this section, we compare the MUDFISH and SUSHSYFISH non-interactive zero-knowledge proof systems to existing methods for proving knowledge of a solution to the MQ or PKP problem. We compare to MQDSS [14] and PKP-DSS [10] that are dedicated proof systems for MQ and PKP respectively, and we compare to Ligero [2] and Aurora [6], which are generic ZK-proof systems capable of proving knowledge of a witness for any NP statement. To compare with generic ZK systems we construct a verification circuit with a minimal number of multiplication gates (since linear gates are for free). For the MQ problem, the verification circuit just evaluates the multivariate system, which requires $n(n+1)/2$ secret multiplications. Using a permutation network [37], we can encode a permutation as a sequence of bits, where each bit controls if a switch in the network is active or not. With this representation, we can verify if a permutation is a solution of a PKP problem with a small number of non-linear gates. If the permutation network has k switches the verification can be done with $2k$ non-linear gates; k multiplications for applying the k switches and an additional k multiplications to verify that the witness consists of bits. Table 4 and 5 show that our proof systems have significantly lower proof sizes compared to existing solutions.

sec. level	Parameters $\mathbb{F}, n$	Circuit Size	Proof Size (KB) MQDSS	Ligero	Aurora	**Mudfish**
128	GF(4), 88	3916	40	199	59	**14**
192	GF(4), 128	8256	43	421	90	**33**
256	GF(4), 160	12880	154	721	358	**56**

Table 4. Comparison of proof sizes of various ZK-proof systems for proving knowledge of a solution of an MQ instance. For the MQDSS system the number of iterations is 315, 478 and 637 respecively. At security level λ, the hashes and commitments are 2λ bits long. The parameter choices do not compensate for the non-tightness of the Fiat-Shamir transform, instead they only guarantee λ bits of soundness for the interactive version of the protocols.

sec.	Parameters	Circuit	Proof Size (KB)			
level	$\mathbb{F}, n, m$	Size	PKP-DSS	Ligero	Aurora	**Sushsyfish**
128	GF(997), 61, 28	606	20	251	46	**12**
192	GF(1409), 87, 42	964	43	385	88	**27**
256	GF(1889), 111, 55	1300	77	539	239	**48**

Table 5. Comparison of proof sizes of various ZK-proof systems for proving knowledge of a solution of a PKP instance.

9 Zero Knowledge proofs for lattice statements

Stern's zero-knowledge protocol has been used extensively in lattice-based cryptography because it can be used to prove a wide variety of statements. It has been used to construct, among other things, identity-based identification schemes, group signatures (with verifier local revocation), logarithmic size ring signatures and group encryption [28, 25, 27, 26]. The way this works is to transform the lattice statement into an instance of the IPKP problem, in such a way that from a solution to the IPKP instance one can efficiently derive a witness for the lattice statement and conversely, that given a witness for the statement one can efficiently compute a solution to the IPKP instance. Then, one just uses Stern's protocol to prove knowledge of a solution to the IPKP instance, which is equivalent to proving knowledge of a witness of the lattice statement. However, it is often the case that witnesses for the lattice statement correspond to IPKP solutions that lie in a certain subgroup $H \subset S_n$. If this is the case, then the prover needs to prove that he knows a solution π to the IPKP instance subject to $\pi \in H$. Luckily, Stern's protocol (and as we will see also our IPKP proof) can be easily adapted to prove knowledge of an IPKP solution that lies in any subgroup H (assuming one can sample uniformly from H and efficiently verify membership of H).

In the remainder of this section, we prove that our IPKP proof can handle proving that a solution lies in a subgroup $H \subset S_n$, which implies that we can improve all the applications of Sterns protocol by replacing Stern's proof by our more efficient protocol. Then, we will focus on proving knowledge of a solution to the inhomogeneous SIS problem. We briefly illustrate how the ISIS problem can be embedded into IPKP with the decomposition-extension technique of ling et al. [28]. Then, we compare the efficiency of our IPKP proof against the efficiency of Stern's protocol for proving knowledge of a solution of an ISIS problem. Finally, we compare our approach to some recent works that use different techniques to prove knowledge of a solution of an ISIS instance.

9.1 Generalizing to Subgroup IPKP

It is trivial to generalize the protocol of Sect. 5 to prove knowledge of a solution π of an IPKP instance with the additional constraint that π lies in a subgroup $H \subset S_n$, assuming that one can efficiently sample uniformly from H and efficiently test if a group element is a member of H. The only modification required is that the prover now samples σ from H instead of from S_n and that the verifier checks that ρ lies in H.

Theorem 5. *The modified version of the protocol for IPKP of Sect. 5 is a sigma protocol with helper as in Definition 3 with challenge space* $\mathbb{F}_q$.

Proof. **Completeness.** If π is a solution of the IPKP instance, then since the unmodified protocol is complete, the verifier will accept a transcript unless the additional check that ρ lies in H fails. However, if $\pi \in H$, then also $\rho = \pi\sigma^{-1}$ lies in H, because σ is sampled from H. Therefore, the verifier will accept with probability 1 if π is a solution to the SIPKP problem.

2-Special Soundness. The extractor from the security proof of the IPKP proof system extracts $\rho\sigma$, which is a solution to the IPKP problem. We only need to show that $\rho\sigma \in H$. The verifier only accepts if $\rho \in H$, and we know that $\sigma \in H$, because it is sampled from H by the honest helper. Therefore the extracted solution to the IPKP solution is also a solution to the SIPKP problem.

Honest-Verifier Zero-Knowledge. The proof is the same as in the proof of Theorem 2, except that the simulator samples τ uniformly from H instead of from S_n.

Remark 1. The proof of Theorem 2 does not use any specific properties of the action of S_n apart from the property that $\mathbf{v}_\sigma + \mathbf{w}_\sigma = (\mathbf{v}+\mathbf{w})_\sigma$, which is required for correctness. Therefore, it is clear that the proof system generalizes to any representation of a finite group G on $\mathbb{F}_q^n$. In particular, we can also consider the group of signed permutations with it natural representation on $\mathbb{F}_q^n$.

9.2 Embedding ISIS into IPKP.

To illustrate the embedding first suppose that $(\mathbf{A}, \mathbf{t}) \in \mathbb{F}_q^{m \times n} \times \mathbb{F}_q^m$ is an instance of the ISIS problem where a solution is a vector $\mathbf{s} \in \mathbb{F}_q^n$ such that $\mathbf{As} = \mathbf{t}$ and the coefficients of $\mathbf{s}$ are 0 or 1. In that case we define the extended matrix $\mathbf{A}' = \begin{pmatrix}\mathbf{A} & \mathbf{0}_{m\times n}\end{pmatrix}$ and a vector $\mathbf{v} \in \mathbb{F}_q$ whose first n entries are 1 and whose last n entries are equal to 0. Then finding a solution to the ISIS instance $(\mathbf{A}, \mathbf{t})$ is equivalent to finding a solution to the IPKP instance $(\mathbf{A}', \mathbf{v}, \mathbf{t})$: Given a solution $\mathbf{s}$ to the ISIS instance it is trivial to find a permutation π such that the first half of $\mathbf{v}_\pi$ equals $\mathbf{s}$, which is then a solution to the IPKP instance. Conversely, if π is a solution to the IPKP instance, then the first half of $\mathbf{v}_\pi$ is a solution to the ISIS instance. Therefore, proving knowledge of π is equivalent to proving knowledge of $\mathbf{s}$.

To improve the efficiency of the proof system we can restrict the IPKP solutions to the subgroup of S_{2n} generated by the transpositions $(i \quad i+n)$ for $0 \le i < n$. This approach reduces the proof size because elements of the subgroup can be represented with only n bits, rather than the $\log_2((2n)!) \approx 2n\log_2(2n)$ bits required to represent an arbitrary permutation of $2n$ elements.

More generally, if the coefficients of $\mathbf{s}$ are required to lie in a range of size B, one can use the decomposition-extension technique [28] to transform an instance of the ISIS problem into an equivalent instance of the IPKP with a matrix $\mathbf{A}'$ with $2n\lceil\log_2 B\rceil$ columns [28]. Moreover, we can restrict to a subgroup of size $2^{2\lceil\log_2 B\rceil}$ to reduce the proof size.

9.3 Concrete examples and comparison to previous works.

To compare the concrete efficiency of our work with the recent work of Bootle at al [11]. and Baum and Nof [5] we apply our proof system to the following two SIS parameters sets:

1. $q \approx 2^{32}, m = 512, n = 2048, \beta = 1$: This is the parameter set considered by Bootle et al [11]. This parameter set is relevant for FHE schemes and group signature schemes.
2. $q \approx 2^{61}, m = 1024, n = 4092,$ binary solution : This is one of the parameter sets considered by Baum and Nof. [5], they claim that this parameter set is relevant for applications such as somewhat homomorphic encryption.

Let $\mathbf{A} \in \mathbb{F}_q^{512\times 2048}$ be an instance of the SIS problem from the first parameter set, define the matrix $\mathbf{A}' = \begin{pmatrix}\mathbf{A}\ \mathbf{0}_{512\times 2048}\end{pmatrix}$ and let $\mathbf{v} \in \{0,1\}^{4096}$ be the vector whose first 2048 entries are equal to 1 and whose remaining 2048 entries are equal to 0. Then finding a solution to the SIS instance $\mathbf{A}$ is equivalent to finding a solution to the generalized PKP instance that asks to find a signed permutation π such that $\mathbf{A}'\mathbf{v}_\pi = \mathbf{0}$. Moreover, this still holds if we restrict the solutions of the PKP problem to lie in the subgroup H generated by sign swaps and the transpositions $\{(i \quad i+2048) |$ for i from 1 to $2048\}$. This subgroup has 8^{2048} elements, and we can represent each element by $3 * 2048$ bits.

Therefore, to prove knowledge of a short integer solution it suffices to prove knowledge of a signed permutation π in H such that $\mathbf{A}'\mathbf{v}_\pi = 0$. We choose parameters $\tau = 14, M = 4040, q' = 2^{10}$ to achieve a soundness error less than 2^{-128}. The proof size is dominated by the vectors and signed permutations in the proof, of which there is one per execution. A vector can be represented with $4069 \log_2(q)$ bits and each permutation in H can be represented with $2048 * 3$ bits. Therefore the total proof size is roughly equal to

$$14 \cdot (4069 \cdot 32 + 2048 \cdot 3) \text{ bits } = 233 \text{ KB}\,.$$

Observe that (in a field of characteristic > 2) if $\overline{1}$ is the vector with a 1 in each entry, then

$$A\mathbf{s} = \mathbf{t} \iff A(2\mathbf{s} - \overline{\mathbf{1}}) = 2\mathbf{t} + A\overline{\mathbf{1}}\,,$$

which means that binary SIS is equivalent to a SIS instance where the entries of $\mathbf{s}$ are restricted to $\{-1, 1\}$. Therefore, for the second parameter set, we can embed the binary SIS problem into a generalized PKP problem of the form $\mathbf{A}\overline{\mathbf{1}}_\pi = \mathbf{t}'$ with π in the group with 2^{4092} elements generated by sign flips. If we again pick $\tau = 14, M = 4040, q' = 2^{10}$ to achieve a soundness error less than 2^{-128} the total proof size is approximately

$$14 \cdot (4092 \cdot 61 + 4092) \text{ bits } = \text{ } 444 \text{ KB}$$

Comparison to previous works. Table 6 makes a comparison of the proof sizes of our proof system with that of previous works. First of all, an application of Stern's protocol to the generalized PKP problems derived from the two parameter sets results in proofs of 2.3 MB and 4.3 MB respectively. This is an order of magnitude larger than our proof system for both parameter sets. The work of Bootle et al. [11] uses algebraic techniques rather than combinatorial ones and achieves a proof size of 384 KB for the first parameter set, which is 65% larger than our proofs.

The proof system of Baum and Nof uses MPC-in-the-head techniques and has a proof size of 4.0 MB for the second parameter set. This is almost an order of magnitude larger than our proofs. Baum and Nof also include timings of the implementation of their protocol. An implementation with 80 bits of statistical security for the second SIS parameter takes 2.4 seconds, with a proof size of 7.5 MB. (Note that this proof size is larger than for their 128 bits variant, because this choice was optimized for speed rather than proof size.) If we choose the parameters for our proof scheme as $q' = 2^4, M = 149, \tau = 23$ to reach 80 bits of statistical security and optimize for speed, our proof size would be 1.4 MB (still 5 times smaller). Extrapolating from our SUSHSYFISH measurements, we claim that with these parameter choices our proof system will be significantly faster than the system of Baum and Nof.

Compared to the generic sub-linear Zero-Knowledge systems Ligero and Aurora [6] our proof systems are asymptotically worse, and for the "large" examples in Table 6 aiming at applications such as FHE we also perform significantly worse in terms of concrete proof sizes. However, for smaller applications, such as proving knowledge of a secret key that corresponds to a MLWE-encryption public key. ($q \approx 2^{13}, n = 1024, m = 512, \beta = 3$) we expect our proof size to be smaller than those of Ligero and similar to those of Aurora. Moreover, an important advantage of our proof system, as well as Stern's protocol and the method of Baum and Nof is that they do not require $\mathbb{F}_q$ (or a field extension thereof) to be NTT friendly, this is in contrast to Ligero, Aurora and the work of Bootle et al..

	$q = 2^{32}$, $m = 512, n = 2048$	$q = 2^{61}$, $m = 1024, n = 4096$
Ours	233 KB	444 KB
Stern [34, 28]	2.3 MB	4.3 MB
Bootle et al. [11]	384 KB	/
Baum and Nof [5]	/	4.0 MB
Aurora [6]	71 KB	71 KB
Ligero [2]	157 KB	200 KB

Table 6. Proof sizes of various protocols for our two SIS parameter sets aiming at 128 bits of security. The hashes and commitments are 256 bits long. The parameter choices do not compensate for the non-tightness of the Fiat-Shamir transform, instead they only guarantee 128 bits of soundness for the interactive version of the protocols.

The work of Del Pino et al. [31] uses so-called bulletproofs to achieve much smaller proof sizes for SIS (for example 1.25 KB for $q \approx 2^{13}, m = 2048, n = 4096$) at the cost of longer running times. However, one cannot make a direct comparison with our work and the other works in Table 6, because bulletproofs rely on the discrete logarithm problem and are hence not post-quantum secure.

Also, there has been a lot of work on “relaxed” proofs for SIS, where the extractor does not output the witness that is known to the prover, but instead some other solution that is somewhat short, but bigger than the witness [29, 4]. For some applications, such as post-quantum signatures [29], this is not a problem, but for other applications, exact proofs are required.

Acknowledgements I would like to thank Simona Samardjiska for helping estimate the complexity of the Joux-Vitse algorithm and Muthu Venkitasubramaniam and Nicholas Ward for providing proof sizes of the Ligero and Aurora proof systems respectively. I would also like to thank the anonymous reviewer for the suggestion to generalize PKP to signed permutations. This work was supported in part by the Research Council KU Leuven: C16/15/058. In addition, this work was supported by the European Commission through the Horizon 2020 research and innovation programme under grant agreement H2020-DS-LEIT-2017-780108 FENTEC, by the Flemish Government through FWO SBO project SNIPPET S007619N and by the IF/C1 on Cryptanalysis of post-quantum cryptography. Ward Beullens is funded by an FWO fellowship.

References

1. Ajtai, M.: Generating hard instances of lattice problems. In: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing. pp. 99–108. ACM (1996)
2. Ames, S., Hazay, C., Ishai, Y., Venkitasubramaniam, M.: Ligero: Lightweight sublinear arguments without a trusted setup. In: Proceedings of the 2017 ACM Sigsac Conference on Computer and Communications Security. pp. 2087–2104 (2017)
3. Baritaud, T., Campana, M., Chauvaud, P., Gilbert, H.: On the security of the permuted kernel identification scheme. In: Annual International Cryptology Conference. pp. 305–311. Springer (1992)
4. Baum, C., Bootle, J., Cerulli, A., Del Pino, R., Groth, J., Lyubashevsky, V.: Sublinear lattice-based zero-knowledge arguments for arithmetic circuits. In: Annual International Cryptology Conference. pp. 669–699. Springer (2018)
5. Baum, C., Nof, A.: Concretely-efficient zero-knowledge arguments for arithmetic circuits and their application to lattice-based cryptography. Tech. rep., Cryptology ePrint Archive, Report 2019/532, 2019. https://eprint. iacr. org ...
6. Ben-Sasson, E., Chiesa, A., Riabzev, M., Spooner, N., Virza, M., Ward, N.P.: Aurora: Transparent succinct arguments for R1CS. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 103–128. Springer (2019)
7. Bernstein, D.: The djbsort software library for sorting arrays of integers or floating-point numbers in constant time. https://sorting.cr.yp.to/

8. Bettale, L., Faugere, J.C., Perret, L.: Hybrid approach for solving multivariate systems over finite fields. Journal of Mathematical Cryptology 3(3), 177–197 (2009)
9. Beullens, W.: FISH. https://github.com/WardBeullens/FISH (2019)
10. Beullens, W., Faugère, J.C., Koussa, E., Macario-Rat, G., Patarin, J., Perret, L.: Pkp-based signature scheme. Cryptology ePrint Archive, Report 2018/714 (2018), https://eprint.iacr.org/2018/714
11. Bootle, J., Lyubashevsky, V., Seiler, G.: Algebraic techniques for short (er) exact lattice-based zero-knowledge proofs. In: Annual International Cryptology Conference. pp. 176–202. Springer (2019)
12. Chase, M., Derler, D., Goldfeder, S., Orlandi, C., Ramacher, S., Rechberger, C., Slamanig, D., Zaverucha, G.: Post-quantum zero-knowledge and signatures from symmetric-key primitives. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. pp. 1825–1842. ACM (2017)
13. Chen, M.S., Hülsing, A., Rijneveld, J., Samardjiska, S., Schwabe, P.: From 5-pass MQ-based identification to MQ-based signatures. In: Cheon, J.H., Takagi, T. (eds.) Advances in Cryptology – Asiacrypt 2016. Lecture Notes in Computer Science, vol. 10032, pp. 135–165. Springer-Verlag Berlin Heidelberg (2016), https://eprint.iacr.org/2016/708
14. Chen, M.S., Hülsing, A., Rijneveld, J., Samardjiska, S., Schwabe, P.: MQDSS-submission to the nist post-quantum cryptography project (2017)
15. Don, J., Fehr, S., Majenz, C., Schaffner, C.: Security of the Fiat-Shamir transformation in the Quantum Random-Oracle Model. In: Annual International Cryptology Conference. pp. 356–383. Springer (2019)
16. Fehr, S.: Classical proofs for the quantum collapsing property of classical hash functions. In: Theory of Cryptography Conference. pp. 315–338. Springer (2018)
17. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Conference on the Theory and Application of Cryptographic Techniques. pp. 186–194. Springer (1986)
18. Georgiades, J.: Some remarks on the security of the identification scheme based on permuted kernels. Journal of Cryptology 5(2), 133–137 (1992)
19. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SIAM Journal on computing 18(1), 186–208 (1989)
20. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Zero-knowledge from secure multiparty computation. In: Proceedings of the thirty-ninth annual ACM symposium on Theory of computing. pp. 21–30. ACM (2007)
21. Jaulmes, É., Joux, A.: Cryptanalysis of PKP: a new approach. In: International Workshop on Public Key Cryptography. pp. 165–172. Springer (2001)
22. Joux, A., Vitse, V.: A crossbred algorithm for solving boolean polynomial systems. In: International Conference on Number-Theoretic Methods in Cryptology. pp. 3–21. Springer (2017)
23. Kales, D., Zaverucha, G.: Forgery attacks on mqdssv2. 0 (2019)
24. Katz, J., Kolesnikov, V., Wang, X.: Improved non-interactive zero knowledge with applications to post-quantum signatures. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. pp. 525–537. ACM (2018)

25. Langlois, A., Ling, S., Nguyen, K., Wang, H.: Lattice-based group signature scheme with verifier-local revocation. In: International Workshop on Public Key Cryptography. pp. 345–361. Springer (2014)
26. Libert, B., Ling, S., Mouhartem, F., Nguyen, K., Wang, H.: Zero-knowledge arguments for matrix-vector relations and lattice-based group encryption. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 101–131. Springer (2016)
27. Libert, B., Ling, S., Nguyen, K., Wang, H.: Zero-knowledge arguments for lattice-based accumulators: logarithmic-size ring signatures and group signatures without trapdoors. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 1–31. Springer (2016)
28. Ling, S., Nguyen, K., Stehlé, D., Wang, H.: Improved zero-knowledge proofs of knowledge for the ISIS problem, and applications. In: International Workshop on Public Key Cryptography. pp. 107–124. Springer (2013)
29. Lyubashevsky, V.: Fiat-shamir with aborts: Applications to lattice and factoring-based signatures. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 598–616. Springer (2009)
30. Patarin, J., Chauvaud, P.: Improved algorithms for the permuted kernel problem. In: Annual International Cryptology Conference. pp. 391–402. Springer (1993)
31. del Pino, R., Lyubashevsky, V., Seiler, G.: Short discrete log proofs for FHE and Ring-LWE ciphertexts. In: Public Key Cryptography. pp. 344–373. Springer (2019)
32. Sakumoto, K., Shirai, T., Hiwatari, H.: Public-key identification schemes based on multivariate quadratic polynomials. In: Annual Cryptology Conference. pp. 706–723. Springer (2011)
33. Shamir, A.: An efficient identification scheme based on permuted kernels. In: Conference on the Theory and Application of Cryptology. pp. 606–609. Springer (1989)
34. Stern, J.: A new identification scheme based on syndrome decoding. In: Annual International Cryptology Conference. pp. 13–21. Springer (1993)
35. Unruh, D.: Non-interactive zero-knowledge proofs in the quantum random oracle model. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 755–784. Springer (2015)
36. Unruh, D.: Computationally binding quantum commitments. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 497–527. Springer (2016)
37. Waksman, A.: A permutation network. Journal of the ACM (JACM) 15(1), 159–163 (1968)

Curriculum Vitae

Ward Beullens obtained his BSc in physics and BSc in mathematics from KU Leuven in 2015. For his Master Thesis work, he conducted research projects under the guidance of Alan Szepieniec and Professor Bart Preneel. After obtaining his MSc in mathematics from KU Leuven in 2017, he joined the COSIC research group in pursuit of a Ph.D. degree, under the supervision of professors Bart Preneel and Frederik Vercauteren, and funded by the Fund for Scientific Research Flanders (FWO). During the fall of 2018, he visited NIST, the US National Institute of Standards and Technology, to study the security of submissions to the NIST post-quantum cryptography standardization project.

FACULTY OF ENGINEERING SCIENCE
DEPARTMENT OF ELECTRICAL ENGINEERING
imec-COSIC
Kasteelpark Arenberg 10, box 2452
B-3001 Leuven
ward.beullens@esat.kuleuven.be
www.cosic.be