Solving the Hidden Number Problem for CSIDH and CSURF via Automated Coppersmith

Jonas Meers and Julian Nowakowski

Ruhr-University Bochum, Bochum, Germany
{jonas.lehmann-c6j,julian.nowakowski}@rub.de

Abstract. We define and analyze the Commutative Isogeny Hidden Number Problem which is the natural analogue of the Hidden Number Problem in the CSIDH and CSURF setting. In short, the task is as follows: Given two supersingular elliptic curves E_A, E_B and access to an oracle that outputs some of the most significant bits of the CDH of two curves, an adversary must compute the shared curve $E_{AB} = \mathsf{CDH}(E_A, E_B)$.
We show that we can recover E_{AB} in polynomial time by using Coppersmith's method as long as the oracle outputs $\frac{13}{24} + \varepsilon \approx 54\%$ (CSIDH) and $\frac{31}{41} + \varepsilon \approx 76\%$ (CSURF) of the most significant bits of the CDH, where $\varepsilon > 0$ is an arbitrarily small constant. To this end, we give a purely combinatorial restatement of Coppersmith's method, effectively concealing the intricate aspects of lattice theory and allowing for near-complete automation. By leveraging this approach, we attain recovery attacks with ε close to zero within a few minutes of computation.

Keywords: Coppersmith, Isogenies, CSIDH, CSURF, Hidden Number Problem

1 Introduction

The Hidden Number Problem (HNP) introduced by Boneh and Venkatesan [5] asks to compute a hidden number α given many tuples $(t_i, \mathsf{MSB}_k(\alpha \cdot t_i \mod p))$ for randomly chosen $t_i \in \mathbb{Z}_p^*$. Here, we denote by $\mathsf{MSB}_k(x)$ the k most significant bits of x. One of the applications of the hidden number problem is the assessment of the bit security of the Diffie-Hellman key exchange over $\mathbb{Z}_p^*$. More concretely, the task can be rephrased as follows: compute the shared Diffie-Hellman key $g^{ab} = \mathsf{CDH}(g^a, g^b) \in \mathbb{Z}_p^*$ given access to an oracle $\mathcal{O}_{\mathsf{MSB}_k}$ that on input $h \in \mathbb{Z}_p^*$ outputs the k most significant bits of $\mathsf{CDH}(g^a, h)$. The famous result by Boneh and Venkatesan states that one can recover g^{ab} in polynomial time if $k \geq \sqrt{\log p}$. Therefore, the $\sqrt{\log p}$ most significant bits of the shared key g^{ab} are as hard to compute as the whole key. The existence of the oracle $\mathcal{O}_{\mathsf{MSB}_k}$ is typically motivated by side-channel attacks and it has recently been shown that such oracles exist in practice [41,35]. Furthermore, the hidden number problem can be used to cryptanalyze ECDSA, Intels Software Guard Extensions (SGX), DSA and qDSA [2,6,17,19,37,47].

The seminal result by Boneh and Venkatesan inspired many follow-up works that investigated different variants of the hidden number problem, for example in the context of Elliptic Curve Diffie-Hellman [53,44,26,4]. As it turns out, the Elliptic Curve Hidden Number Problem (EC-HNP) is already much harder to solve and requires different techniques. In particular, Boneh, Halevi and Howgrave-Graham propose in [3] to use *Coppersmith's method* [14,13] to solve EC-HNP for $k \geq 0.98 \log p$ and curves defined over $\mathbb{F}_p$. More recently, this approach was further improved, making it feasible to solve EC-HNP for $k \geq \frac{1}{d+1} \log p$ and any fixed $d > 0$ [54].[1] Here the key ingredient is (a very involved variant of) Coppersmith's method.

The recent advent of quantum computers completely bypasses the bit security statements of the (elliptic curve) Diffie-Hellman key exchange since the discrete logarithm problem for groups can be solved in quantum polynomial time due to Shor's algorithm [45]. To thwart this issue, many post-quantum secure alternatives have been proposed. One popular approach is based on *isogenies* which are rational maps between (supersingular) elliptic curves. In some settings, isogenies give rise to cryptographic group actions in the sense of [1] which behave very similarly to exponentiation in prime order groups. Due to this (syntactical) similarity, many protocols and results from the Diffie-Hellman context have been adapted to the isogeny setting (for example [28,20,29,55]). However, this is not the case for the bit security of isogeny based key exchanges. One of the few results in that area studies the bit security of the SIDH key exchange and states that computing one component of the secret j-invariant of a curve is as hard as computing both components [22]. Due to the recent devastating attacks on SIDH [8,31,40], however, the statement about its bit security is now obsolete. Apart from SIDH there still exist (non-interactive) key exchanges based on isogenies that are still believed to be post-quantum secure. The most prominent examples are CSIDH [10] and CSURF [7], both of which are based on the group action of fractional ideals on the set of supersingular elliptic curves over $\mathbb{F}_p$. These key exchanges are not affected by the attacks on SIDH, yet very little is known about their bit security.

1.1 Our Contributions

In this work, we close this gap and analyze the hardness of the Commutative Isogeny Hidden Number Problem (CI-HNP) for CSIDH and CSURF, which can be informally stated as follows:

Commutative Isogeny Hidden Number Problem (informal). Given two public curves E_A, E_B defined over $\mathbb{F}_p$ and access to an oracle $\mathcal{O}_{\mathsf{MSB}_k}$ that on input two elliptic curves E, E' outputs $\mathsf{MSB}_k(\mathsf{CDH}(E, E'))$, recover the shared curve $E_{AB} = \mathsf{CDH}(E_A, E_B)$.

[1] Note that if d is not fixed the runtime of the algorithm is in fact super-exponential in d.

Solving CI-HNP. Our first major contribution is a (heuristic) polynomial time algorithm based on Coppersmith's method that recovers the shared curve E_{AB} for $k = (\frac{13}{24} + \varepsilon)n$ (CSIDH) and $k = (\frac{31}{41} + \varepsilon)n$ (CSURF), where $n = \log p$ and $\varepsilon > 0$ is an arbitrarily small constant. We remark that our results do not yield an *unconditional* bit security statement for the respective non-interactive key exchanges due to the heuristic nature of Coppersmith's method. Nevertheless, our result implies that (under some constraints) computing the $\frac{13}{24}n$ (CSIDH) and $\frac{31}{41}n$ (CSURF) most significant bits of the shared curve E_{AB} is as hard as solving CDH and quantumly as hard as solving DLOG due to the quantum equivalence of the latter two assumptions [52].

Automated Coppersmith. As our second major contribution, we give a significantly simplified reformulation of Coppersmith's method. This allows us to automate Coppersmith's method almost entirely, and to easily apply it to CI-HNP. This is in stark contrast to almost all previous Coppersmith-type results, which typically required highly involved lattice constructions that had to be fine-tuned using ad-hoc techniques. (See, e.g., May's recent survey [32] and the references therein.) Our approach, on the other hand, only requires to specify which monomials we want to include in our lattice basis. For any given set of monomials our approach then automatically (and efficiently) constructs the corresponding *optimal* lattice.

We also give a simple automated strategy for selecting these monomials. While this strategy might not always yield optimal results, it performs well in practice, and allows us to derive our bounds for CI-HNP. Furthermore, it enables another interesting application: For any given system of polynomial equations our algorithm can (under some reasonable heuristics) *automatically* derive upper bounds on the size of the roots that can be recovered by Coppersmith's method – a process that prior to our work involved a lot of manual effort. Our reformulation of Coppersmith's method is not specific to the application at hand and might therefore be of independent interest.

Implementation. As our third contribution we provide an efficient open source implementation of our automated variant of Coppersmith's method in SageMath. The source code is available at

`https://github.com/juliannowakowski/automated-coppersmith`.

Using this implementation, we run our algorithm for CI-HNP on cryptographically sized instances with bitsize $n = 512, 1024, 1792$. Our experimental results verify the correctness of our heuristic algorithm, and show that we come close to the asymptotic bounds of $k = \frac{13}{24}n \approx 0.542n$ and $k = \frac{31}{41}n \approx 0.756n$ in a matter of minutes.

1.2 Technical Details and Related Work

Similar to the case of EC-HNP we use Coppersmith's method to recover the least significant bits of the Montgomery coefficient of the shared curve E_{AB}. To this end, we define polynomial relations between the Montgomery coefficient

of E_{AB} and the Montgomery coefficients of its d-isogenous neighboring curves, essentially replicating the behaviour of the *modular polynomials* but for the case of Montgomery coefficients. We then embed the partial information from the oracle $\mathcal{O}_{\mathsf{MSB}_k}$ in the coefficients of these polynomials by querying $\mathcal{O}_{\mathsf{MSB}_k}$ on specific input. As a consequence, we can construct a system of polynomial equations that has a common small root in the Montgomery coefficient of the curve E_{AB}. In a last step, this small root is found by Coppersmith's method.

Comparison with HNP and EC-HNP. In their original work Boneh and Venkatesan use lattice reduction techniques to solve HNP over $\mathbb{Z}_p^*$ [5]. More specifically, given many oracle queries one derives an underdetermined system of *linear* equations that is subsequently encoded into a lattice. By solving a closest vector problem in the lattice one obtains the secret value. With EC-HNP it is already much harder to implement this approach as the system of equations is inherently *nonlinear*. In fact, each query to the oracle results in a bivariate polynomial of total degree 3. The secret value is then encoded in a common small root of these polynomials, and recovered via Coppersmith's method [54]. Nevertheless, both HNP and EC-HNP have in common that one can get arbitrarily many equations by querying the oracle $\mathcal{O}_{\mathsf{MSB}_k}$ as many times as needed. Phrased differently, the system of polynomial equations – while still being underdetermined – can be made arbitrarily large. Furthermore, each polynomial in the system of equations has the same *shape*.

Unfortunately, in the case of CI-HNP neither of the two properties hold. In both the CSIDH and CSURF settings, each curve has for any given degree d at most two d-isogenous neighbours defined over $\mathbb{F}_p$. Hence, if we wish to make many such oracle queries we necessarily have to use isogenies of higher degree, which in turn result in high-degree polynomial relations. Therefore, we are left with a choice: either have few polynomials of low degree or have many polynomials with very high degree. Additionally, by changing the degree of the isogeny one obtains polynomials of a different *shape*, making optimizations very challenging.

Recovery Rates. Curiously, the recovery rates between CSIDH and CSURF differ quite significantly. It turns out that the reason for this is an order-3 subgroup of the ideal class group $cl(\mathcal{O})$ in the CSIDH setting that is not present in CSURF. Subsequently, in the CSIDH setting we are able to construct *more* polynomial relations which have *smaller* degree compared to the CSURF setting. Since Coppersmith's method performs best for polynomials with low degree, this results in a better recovery rate. It is worth mentioning that in the context of analyzing the security of the CSIDH key exchange the same order-3 subgroup is also responsible for reducing the security of the key exchange by a factor of $\frac{1}{3}$ [11,38].

Comparison with SIDH. The Isogeny Hidden Number Problem has been considered before by Galbraith, Petit, Shani and Ti in the context of SIDH [22]. However, their approach only applies to SIDH due to the fact that in this setting the resulting polynomial equations are defined over $\mathbb{F}_{p^2}$. If we let $\mathbb{F}_{p^2} = \mathbb{F}_p(\theta)$ where $\theta^2 \in \mathbb{F}_p$ denotes some quadratic non-residue, then any equation $f(j) = 0$

over $\mathbb{F}_{p^2}$ results in *two* equations over $\mathbb{F}_p$: for $f(j) = f_{\mathsf{real}}(j) + f_{\mathsf{im}}(j) \cdot \theta = 0$, we must have $f_{\mathsf{real}}(j) = 0$ and $f_{\mathsf{im}}(j) = 0$ simultaneously. This trick in combination with the modular polynomial allowed the authors of [22] to build a system of two polynomial equations in two unknowns which can be solved exactly. Subsequently the authors were able to recover one component of the secret j-invariant given an oracle that returns the other component.

This approach is not applicable to CSIDH and CSURF as in this context the polynomial relations are necessarily defined over $\mathbb{F}_p$. We therefore have to resort back to heavy machinery like Coppersmith's method to solve systems of polynomial equations.

Coppersmith's Method. To solve a system of polynomial equations, Coppersmith's method requires as input a set of well-chosen *shift-polynomials*. Crucially, these shift-polynomials f_i must satisfy several technical constraints imposed by Coppersmith's method while simultaneously minimizing the determinant of a certain matrix. Concretely, the matrix has as entries the coefficient vectors of the f_i. In the process of selecting the f_i a ripple effect can occur where a locally optimal choice of a single f_i leads to an overall larger determinant. We observe, however, that choosing *globally optimal* f_i can be fully automated once we fix the set of monomials over which the f_i are defined. Therefore, the only non-trivial task is choosing a "good" set of monomials. The subsequent optimal construction of the f_i then reduces to a purely combinatorial strategy, somewhat similar to the celebrated *Jochemsz-May* strategy [27]. However, we significantly improve on Jochemsz-May since we can handle *systems* of polynomial equations, whereas their strategy only handles *single* polynomial equations. This is particularly useful for the application at hand as in the case of CI-HNP we must deal with such a system of polynomial equations.

Computing Asymptotic Bounds. Similar to the construction of the shift-polynomials, the task of determining asymptotic upper bounds for Coppersmith's method is typically very time-consuming and has to be performed manually each time a new set of polynomials is considered. Moreover, the proof that a given asymptotic bound holds is oftentimes convoluted. We overcome both issues by combining our automated variant of Coppersmith's method with *polynomial interpolation*. More specifically, given a system of polynomial equations our algorithm determines (under some reasonable heuristics) the size of the largest root that can be recovered. This upper bound may not be optimal with respect to the given system of polynomial equations, but nevertheless serves as a good starting point. We demonstrate the usefulness of this approach in the main body of this paper. In addition, the accompanying proof is easy to verify but crucially relies on the correctness of the heuristic. Fortunately, it appears that from the output of our algorithm it is always possible to extract a rigorous proof of correctness that does not involve the aforementioned heuristic. This task, however, requires manual work.

1.3 Outline of the Paper

The paper is organized as follows: In Section 2 we give some basic preliminaries for CSIDH, CSURF and Coppersmith's method. Our new formulation of Coppersmith's method is described in Section 3 and proven in Section 4. In Section 5 we show how to solve CI-HNP and discuss the quantum hardness of simulating $\mathcal{O}_{\mathsf{MSB}_k}$. In Section 6 we give results on the practical recovery rate of our heuristic algorithm, which experimentally verify its correctness. We conclude our work in Section 7 where we also state some open problems.

2 Preliminaries

We use the notation $x \xleftarrow{\$} \mathcal{X}$ to indicate that x is uniformly sampled from a set $\mathcal{X}$. By $\log n$ we denote the base 2 logarithm of n. For a prime p with $p \equiv 3 \mod 4$ and a square $a \in \mathbb{F}_p$ we further define $\sqrt{a} \in \mathbb{F}_p$ to be the unique square root of a which is itself again a square. It can be computed as $\sqrt{a} = a^{(p+1)/4} \mod p$. For a n-bit prime p and an integer $x \in \mathbb{Z}_p$ we denote by $\mathsf{MSB}_k(x)$ the k most significant bits of x, i.e. the integer t such that $0 \leq x - t \cdot 2^{n-k} < p/2^k$.

2.1 Elliptic Curves and Isogenies

The following facts about isogenies are mostly taken from Silverman [46].

Let $E/\mathbb{F}_p$ be an elliptic curve over a finite field $\mathbb{F}_p$ with p an odd prime. We denote the point at infinity with ∞_E. For an extension field $\mathbb{K} \supseteq \mathbb{F}_p$ we denote the set of $\mathbb{K}$-rational points by $E(\mathbb{K})$. An elliptic curve is called *supersingular* if $\#E(\mathbb{F}_p) = p + 1$ and *ordinary* otherwise.

An isogeny is a morphism $\varphi : E \to E'$ between elliptic curves E, E' such that $\varphi(\infty_E) = \infty_{E'}$. The degree of φ is its degree as a morphism and we call φ *separable* if $p \nmid \deg \varphi$. An isogeny can be expressed as a fraction of polynomials and we call two elliptic curves *isogenous* if there exists an isogeny between them. An isogeny is called an isomorphism if it has an inverse (which may be defined over the algebraic closure of $\mathbb{F}_p$). In that case the inverse is again an isogeny. One can check whether two elliptic curves are isomorphic by comparing their *j-invariant*, which is a simple algebraic expression in the coefficients of the curve equation.

An isogeny from E to itself is called an *endomorphism*. The set $\mathrm{End}(E)$ of endomorphisms of E (defined over the algebraic closure) forms a ring under addition and composition and is thus called the *endomorphism ring*. We denote by $\mathrm{End}_p(E)$ the subring defined over $\mathbb{F}_p$, which is an order in the imaginary quadratic field $\mathbb{Q}(\sqrt{-p})$ if E is supersingular.

Any isogeny $\varphi : E \to E'$ is automatically a group homomorphism from E to E' and as such its kernel is a finite subgroup of E. In the case where φ is separable we have that $\deg \varphi = \# \ker \varphi$. Conversely, any finite subgroup $G \subset E$ corresponds to a separable isogeny $\varphi : E \to E'$ with kernel $\ker \varphi = G$ that is unique up to post-composition with an isomorphism. Since E' is uniquely

determined by $\ker \varphi$ (again up to isomorphism), we write $E' = E/G$ from now on. One can compute φ and E/G via Vélus formula [51], which can be evaluated in time polynomial in the size of the kernel.

For an integer n we denote the multiplication-by-n map by $[n]$, which is an endomorphism of E. Its kernel is the *n-torsion subgroup* $E[n] = \{P \in E : [n]P = \infty_E\}$. Another important endomorphism is the *Frobenius endomorphism* π_E, sending $(x, y) \in E$ to $(x^p, y^p) \in E$. In the case where E is supersingular it satisfies $\pi_E \circ \pi_E = -p$, implying that $\mathbb{Z}[\sqrt{-p}] \subseteq \mathrm{End}_p(E)$.

2.2 Group Actions from Isogenies

Currently there exist two popular constructions for an isogeny-based group action, namely CSIDH [10] and CSURF [7]. They mainly differ in the choice of $\mathrm{End}_p(E)$. Indeed, if $p \equiv 3 \mod 4$ and E is supersingular there are two choices for $\mathrm{End}_p(E)$, namely $\mathbb{Z}[\sqrt{-p}]$ and $\mathbb{Z}[(1+\sqrt{-p})/2]$. The following section is mostly based on [7,10], however we also incorporate some recent suggestions related to CSURF stated in [12].

CSIDH. Let $p = 4 \cdot \ell_1 \dots \ell_n - 1$ be a prime where the ℓ_i are small odd primes. Fix the order $\mathcal{O} = \mathbb{Z}[\pi]$, where $\pi = \sqrt{-p}$ is the Frobenius endomorphism. Let $\mathcal{E}\ell\ell_p(\mathcal{O})$ be the set of supersingular elliptic curves E defined over $\mathbb{F}_p$ with endomorphism ring $\mathrm{End}_p(E) \cong \mathcal{O}$ (called the *floor*). The ideal class group $cl(\mathcal{O})$ acts on the set $\mathcal{E}\ell\ell_p(\mathcal{O})$ in the following way: to each $\mathfrak{a} \subseteq \mathcal{O}$ we can associate the subgroup

$$E[\mathfrak{a}] \coloneqq \bigcap_{\varphi \in \mathfrak{a}} \{P \in E : \varphi(P) = \infty_E\} \subseteq E.$$

Here, we view φ as an endomorphism of E through the isomorphism $\mathrm{End}_p(E) \cong \mathcal{O}$.

Theorem 1 (Theorem 4.5 of [43]). *The map*

$$\star : cl(\mathcal{O}) \times \mathcal{E}\ell\ell_p(\mathcal{O}) \to \mathcal{E}\ell\ell_p(\mathcal{O}),$$

sending $([\mathfrak{a}], E)$ *to* $\mathfrak{a} \star E \coloneqq E/E[\mathfrak{a}]$ *is a well-defined free and transitive group action.*

Observe that because $p \equiv -1 \mod \ell_i$ the ideal (ℓ_i) splits in $\mathbb{Z}[\pi]$ as $(\ell_i) = \langle \ell_i, \pi - 1\rangle\langle \ell_i, \pi + 1\rangle$. Additionally, since $\#E(\mathbb{F}_p) = p+1$ each curve in $\mathcal{E}\ell\ell_p(\mathcal{O})$ has an $\mathbb{F}_p$-rational point $P_\rightarrow$ generating a subgroup of order ℓ_i, which corresponds to the ideal $\mathfrak{l}_i = \langle \ell_i, \pi - 1\rangle$. Therefore, the action $\star$ can be computed efficiently for the ideals $\mathfrak{l}_i$ by finding $P_\rightarrow$ and then applying Vélu's formula. A similar reasoning applies to the ideal $\bar{\mathfrak{l}}_i = \langle \ell_i, \pi + 1\rangle$ where the only difference is that the generating point $P_\leftarrow$ of order ℓ_i has its x-coordinate in $\mathbb{F}_p$ but its y-coordinate outside of $\mathbb{F}_p$. Therefore, the CSIDH group action can be evaluated efficiently for ideals of the form $\prod \mathfrak{l}_i^{e_i}$, where the e_i are from a small range $[-B, B]$.

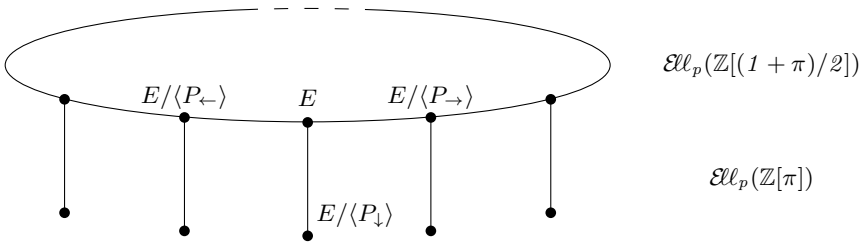

Fig. 1. The 2-isogeny graph for a prime $p \equiv 7 \mod 8$.

For each E on the floor there exists a unique $A \in \mathbb{F}_p$ called the *Montgomery coefficient* such that E is isomorphic to the curve $E_A : y^2 = x^3 + Ax^2 + x$ [10, Proposition 8]. The curve E_A is called the *Montgomery form* of E and we denote by $\mathcal{M}_p(\mathcal{O})$ the set of Montgomery coefficients of curves in $\mathcal{E}\ell\ell_p(\mathcal{O})$. We can now see the group action $\star$ equivalently as a group action

$$\star : cl(\mathcal{O}) \times \mathcal{M}_p(\mathcal{O}) \to \mathcal{M}_p(\mathcal{O}), \tag{1}$$

where we identify each $A \in \mathcal{M}_p(\mathcal{O})$ with the curve E_A. By slight abuse of notation we denote this action by $\star$ as well.

Lastly, we define $E_0 : y^2 = x^3 + x$ to be the starting curve of the group action. Indeed, E_0 has endomorphism ring $\mathbb{Z}[\pi]$ and therefore lives on the floor.

CSURF. Let $p = 4 \cdot \ell_0 \dots \ell_n - 1$ be a prime such that $\ell_0 = 2$ and the ℓ_i are small odd primes for $i > 0$. Fix the order $\mathcal{O} = \mathbb{Z}[(1+\pi)/2]$ where again $\pi = \sqrt{-p}$ is the Frobenius endomorphism and $\mathcal{E}\ell\ell_p(\mathcal{O})$ is the set of supersingular elliptic curves with endomorphism ring $\mathcal{O}$ (which is now called the *surface*). The ideal class group $cl(\mathcal{O})$ acts in a very similar way on $\mathcal{E}\ell\ell_p(\mathcal{O})$. In fact, the action of the ideals $\mathfrak{l}_i$ with $i > 0$ can be evaluated in the same way as in CSIDH. The only difference is that the ideal (2) now splits in $\mathbb{Z}[(1+\pi)/2]$ as $(2) = \langle 2, (\pi-1)/2 \rangle \langle 2, (\pi+1)/2 \rangle$ due to the congruence $p \equiv 7 \mod 8$.

This means that there are two additional ideals $\mathfrak{l}_0$ and $\bar{\mathfrak{l}}_0$ available for the group action. In contrast to the odd degree isogenies, for each $E \in \mathcal{E}\ell\ell_p(\mathcal{O})$ there are now *three* points of order 2 with x-coordinate in $\mathbb{F}_p$. It turns out that $\mathfrak{l}_0$ is generated by a point $P_\rightarrow$ of order 2 whose four halves are all $\mathbb{F}_p$-rational. Similarly, the four halves of the point $P_\leftarrow$ generating $\bar{\mathfrak{l}}_0$ have x-coordinate in $\mathbb{F}_p$ but y-coordinate outside of $\mathbb{F}_p$. The remaining point $P_\downarrow$ of order two has its four halves completely outside of $\mathbb{F}_p$ and quotienting out $\langle P_\downarrow \rangle$ results in a curve on the floor (see Figure 1). In order to compute the action of $\mathfrak{l}_0$ and $\bar{\mathfrak{l}}_0$, one first finds the corresponding point of order 2 and then applies Vélu's formula. In accordance with the literature, we call the isogenies generated by $P_\leftarrow$ and $P_\rightarrow$ *horizontal*, whereas the isogeny generated by $P_\downarrow$ is called *vertical*.

Another difference to CSIDH is that there are now two isomorphic curves in Montgomery form for each $E \in \mathcal{E}\ell\ell_p(\mathcal{O})$ [7, Corollary 1]. To make the choice unique one can choose the curve $E_A : y^2 = x^3 + Ax^2 + x$ such that $A \pm 2$ are

both squares[2] in $\mathbb{F}_p$. As before, let $\mathcal{M}_p(\mathcal{O})$ denote the set of such Montgomery coefficients. We again have a group action

$$\star : cl(\mathcal{O}) \times \mathcal{M}_p(\mathcal{O}) \to \mathcal{M}_p(\mathcal{O}), \tag{2}$$

where $A \in \mathcal{M}_p(\mathcal{O})$ is identified with E_A.

Lastly, we set the starting curve to be $E_{3/\sqrt{2}} : y^2 = x^3 + (3/\sqrt{2})x^2 + x$, which has endomorphism ring $\mathbb{Z}[(1+\pi)/2]$. Note that due to the convention on modular square roots, we also have that

$$\frac{3}{\sqrt{2}} \pm 2 = \frac{1}{\sqrt{2}}(3 \pm 2\sqrt{2}) = \frac{1}{\sqrt{2}}(1 \pm \sqrt{2})^2$$

are both squares.

Remark 1. In [7] it was suggested to identify $E \in \mathcal{E}\ell\ell_p(\mathcal{O})$ with its corresponding *Montgomery*$^-$ *form* $E_A^- : y^2 = x^3 + Ax^2 - x$ as it uniquely represents the isomorphism class of E [7, Proposition 4]. However, this suggestion was later revoked due to slower low-level arithmetics on Montgomery$^-$ curves [12, p. 12]. Additionally, one uses regular Montgomery curves to compute the action of the 2-isogenies anyway [7, Algorithm 1], which is why we choose to work with regular Montgomery curves as well.

2.3 Cryptographic Assumptions and Protocols

The CSIDH and CSURF group action can be used to instantiate a non-interactive key exchange (NIKE) similar to the Hashed Diffie-Hellman key exchange over prime-order groups (see Figure 2). In the Random Oracle Model its passive security relies on the hardness of the following two problems, which go back to Couveignes (who called them Vectorization and Parallelization) [16]. Both definitions apply to the CSIDH and CSURF setting.

Definition 1 (Discrete Logarithm Problem (DLOG)). *Let $E \in \mathcal{E}\ell\ell_p(\mathcal{O})$ be a fixed starting curve and $[\mathfrak{a}] \xleftarrow{\$} cl(\mathcal{O})$. Given the tuple $(E, \mathfrak{a} \star E)$, recover $[\mathfrak{a}]$.*

Definition 2 (Computational Diffie-Hellman Problem (CDH)). *Let $E \in \mathcal{E}\ell\ell_p(\mathcal{O})$ be a fixed starting curve and $[\mathfrak{a}], [\mathfrak{b}] \xleftarrow{\$} cl(\mathcal{O})$. Given the tuple $(E, \mathfrak{a} \star E, \mathfrak{b} \star E)$, compute $\mathfrak{ab} \star E$.*

Remark 2. In the following we leave out the starting curve E as long as there is no ambiguity.

Galbraith et al. showed that for efficiently computable group actions, CDH is equivalent to DLOG in a quantum setting [21]. Their reduction assumes a perfect adversary $\mathcal{A}$ against CDH, i.e. an adversary with success probability 1, which is then used to construct a quantum adversary against DLOG. Moreover,

[2] This actually guarantees that $P_\to = (0, 0)$.

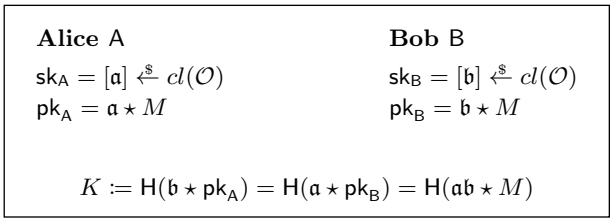

Fig. 2. Non-interactive key exchange based on CSIDH/CSURF where M is the Montgomery coefficient of a fixed starting curve and $\mathsf{H} : \{0,1\}^* \to \{0,1\}^\lambda$ is a hash function.

they assume that the action of a random element $[\mathfrak{a}] \xleftarrow{\$} cl(\mathcal{O})$ can be computed efficiently, which in general is not the case for the CSIDH and CSURF group action. Furthermore, it is currently not known how to sample an element $[\mathfrak{a}]$ uniformly at random for arbitrary parameter sets.

More recently, their result was improved by Montgomery and Zhandry who showed that the equivalence also holds for any adversary $\mathcal{A}$ having a non-negligible success probability [36]. The authors also gave some mild evidence that the equivalence holds for *restricted effective group actions* (of which CSIDH and CSURF are instantiations, see also [1]), but this result only holds for generic adversaries making classical queries to a group action oracle.

In the special case of the CSIDH and CSURF group action, Wesolowski showed that CDH and DLOG are quantumly equivalent under the generalized Riemann hypothesis [52].

2.4 Polynomials

Let $x_1, \ldots, x_k$ be symbolic variables. A *monomial* is a product of the form $x_1^{i_1} \cdot \ldots \cdot x_k^{i_k}$, where $i_1, \ldots, i_k \in \mathbb{N}$. In particular, a product of the form $c \cdot x_1^{i_1} \cdot \ldots \cdot x_k^{i_k}$, where $c \neq 1$, is not a monomial. Let $f(x_1, \ldots, x_k) = \sum_{i_1,\ldots,i_k \in \mathbb{N}} \alpha_{i_1,\ldots,i_k} \cdot x_1^{i_1} \cdot \ldots \cdot x_k^{i_k}$ be a polynomial with coefficients $\alpha_{i_1,\ldots,i_k} \in \mathbb{Z}$. We say that $x_1^{i_1} \cdot \ldots \cdot x_k^{i_k}$ is a *monomial of* f, if $\alpha_{i_1,\ldots,i_k} \neq 0$. If all monomials of f are elements of some set $\mathcal{M}$, then we say that f *is defined over* $\mathcal{M}$. We denote by $\deg(f)$ the *total degree* of f, i.e.,

$$\deg(f) := \max_{\alpha_{i_1,\ldots,i_k} \neq 0} (i_1 + \ldots + i_k).$$

The degree of some finite set of polynomials $\mathcal{F} \subseteq \mathbb{Z}[x_1, \ldots, x_k]$ is defined as

$$\deg(\mathcal{F}) := \max_{f \in \mathcal{F}} \deg(f).$$

The norm of f, denoted $\|f\|$, is defined as the Euclidean norm of its coefficient vector, i.e.,

$$\|f\| := \sqrt{\sum_{i_1,\ldots,i_k \in \mathbb{N}} \alpha_{i_1,\ldots,i_k}^2}.$$

Definition 3. *For a set of polynomials $\mathcal{F} \subset \mathbb{Z}[x_1, \ldots, x_k]$, we define the set of its* integer roots *as*

$$Z_{\mathbb{Z}}(\mathcal{F}) := \left\{ r = (r_1, \ldots, r_k) \in \mathbb{Z}^k \mid \forall f \in \mathcal{F} : f(r) = 0 \right\}.$$

Similarly, for parameters $M, X_1, \ldots, X_k \in \mathbb{N}$, we define the corresponding set of its small modular roots *as*

$$Z_{M,X_1,\ldots,X_k}(\mathcal{F}) := \left\{ r = (r_1, \ldots, r_k) \in \mathbb{Z}^k \middle| \begin{matrix} \forall f \in \mathcal{F} : f(r) \equiv 0 \mod M, \\ \forall j : |r_j| \leq X_j \end{matrix} \right\}.$$

For a finite set $\mathcal{F} = \{f_1, \ldots, f_n\}$, we may abuse notation and write

$$Z_{\mathbb{Z}}(f_1, \ldots, f_n) := Z_{\mathbb{Z}}(\mathcal{F}),$$
$$Z_{M,X_1,\ldots,X_k}(f_1, \ldots, f_n) := Z_{M,X_1,\ldots,X_k}(\mathcal{F}).$$

Definition 4. *Let $\mathcal{M}$ be a set of monomials. A* monomial order (on $\mathcal{M}$) *is a total order $\prec$ on $\mathcal{M}$, that satisfies the following two properties:*

1. *For every $\lambda \in \mathcal{M}$, it holds that $1 \prec \lambda$.*
2. *If $\lambda_1 \prec \lambda_2$, then $\lambda \cdot \lambda_1 \prec \lambda \cdot \lambda_2$ for every monomial $\lambda \in \mathcal{M}$.*

We frequently use the *lexicographic monomial order* $\prec_{\mathsf{lex}}$ .The *leading monomial* of a polynomial f (with respect to some monomial order $\prec$) is the unique monomial λ of f, which satisfies $\lambda' \prec \lambda$ for every monomial λ' of f. The coefficient of the leading monomial is called *leading coefficient*. If the monomial order is clear from the context, we denote by $\mathsf{LM}(f)$ and $\mathsf{LC}(f)$ the leading monomial and the leading coefficient of f, respectively. Notice that for any two polynomials f, g we have

$$\mathsf{LM}(fg) = \mathsf{LM}(f)\mathsf{LM}(g), \tag{3}$$
$$\mathsf{LC}(fg) = \mathsf{LC}(f)\mathsf{LC}(g). \tag{4}$$

If $\mathsf{LC}(f) = 1$, then we say that f is *monic*.

2.5 Lattices

A *(full-rank) lattice* is a set of the form $\mathcal{L}(\mathbf{B}) := \mathbf{B} \cdot \mathbb{Z}^d$, where $\mathbf{B} \in \mathbb{R}^{d \times d}$ is an invertible matrix. We call $\mathbf{B}$ a *basis matrix* of $\mathcal{L}(\mathbf{B})$ and say that $\mathcal{L}(\mathbf{B})$ is the lattice generated by the columns of $\mathbf{B}$. The value d is called the *dimension* of $\mathcal{L}(\mathbf{B})$. The *determinant* of $\mathcal{L}(\mathbf{B})$ is defined as $\det \mathcal{L}(\mathbf{B}) := |\det \mathbf{B}|$. We call two basis matrices $\mathbf{B}_1, \mathbf{B}_2 \in \mathbb{R}^{d \times d}$ *equivalent*, if $\mathcal{L}(\mathbf{B}_1) = \mathcal{L}(\mathbf{B}_2)$. For equivalent basis matrices $\mathbf{B}_1, \mathbf{B}_2$ it holds that $\det \mathcal{L}(\mathbf{B}_1) = \det \mathcal{L}(\mathbf{B}_2)$. The *norm* of a lattice vector $\mathbf{v} \in \mathcal{L}(\mathbf{B})$, denoted $\|\mathbf{v}\|$, is the Euclidean norm.

The famous LLL lattice-reduction algorithm [30] computes on input of a lattice basis $\mathbf{B} = (b_{i,j})_{1 \leq i \leq j \leq d} \in \mathbb{Z}^{d \times d}$ an equivalent basis in time polynomial in d and $\max_{i,j} \log(|b_{i,j}|)$, consisting of relatively short lattice vectors:

Lemma 1. *Let* $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_d)$ *be an LLL-reduced basis of a* d*-dimensional lattice* $\Lambda \subseteq \mathbb{Z}^d$ *and let* $M, m \in \mathbb{N}$*, such that* $\log(M) \geq d \geq m$*. Suppose that*

$$\det(\Lambda) \leq M^{(m-k)d}$$

holds for some $k \leq d$*. Then*

$$\|\mathbf{b}_i\| < \frac{M^m}{\sqrt{d}}$$

holds for every $i = 1, \ldots, k$*.*

A proof for Lemma 1 is given in the full version of the paper.

3 Coppersmith's Method

In this section, we introduce our significantly simplified reformulation of Coppersmith's method. In Section 3.1, we recall the high-level idea behind Coppersmith's method, as well as the heuristic, that is used in almost all Coppersmith-type results. After that, we give in Section 3.2 a purely combinatorial reformulation and show how this allows us to automate Coppersmith's method almost entirely. As an application of our reformulation, we derive in Section 3.3 two new Coppersmith-type bounds, which we use in Section 5 to prove our results for CI-HNP.

3.1 High Level Idea

Suppose we are given a modulus $M \in \mathbb{N}$, polynomials $f_1, \ldots, f_n \in \mathbb{Z}_M[x_1, \ldots, x_k]$ and bounds $X_1, \ldots, X_k \in \mathbb{N}$. If the bounds are sufficiently small (and k is fixed), then Coppersmith's method finds all small modular roots

$$r \in Z_{M,X_1,\ldots,X_k}(f_1, \ldots, f_n)$$

in time polynomial in $\log(M)$.

The main idea behind Coppersmith's method is to transform the system of polynomial equations defined by the f_i's over $\mathbb{Z}_M$ into an efficiently solvable system of equations defined over $\mathbb{Z}$. To this end, Coppersmith's method uses lattice-based techniques to construct k polynomials $h_1, \ldots, h_k \in \mathbb{Z}[x_1, \ldots, x_k]$, such that all small modular roots of the f_i's are also integer roots of the h_i's, i.e.,

$$Z_{M,X_1,\ldots,X_k}(f_1, \ldots, f_n) \subseteq Z_{\mathbb{Z}}(h_1, \ldots, h_k).$$

Given the h_i's, we can efficiently compute all small modular roots r as follows:

In the univariate setting, where $k = 1$, we simply compute all roots of h_1 over $\mathbb{R}$ using standard techniques (such as Newton's method or the Sturm sequence) and then output those that lie in $Z_{M,X_1}(f_1, \ldots, f_n)$. In the multivariate setting, where $k > 1$, we follow a Gröbner basis based approach. Here, we first compute the Gröbner basis of the ideal $\mathfrak{a} := (h_1, \ldots, h_k) \subseteq \overline{\mathbb{Q}}[x_1, \ldots, x_k]$. Assuming that the variety of $\mathfrak{a}$ is zero-dimensional (which is usually the case in practice) we then efficiently obtain $Z_{\mathbb{Z}}(h_1, \ldots, h_k)$ from the Gröbner basis, again using standard techniques. Finally, from $Z_{\mathbb{Z}}(h_1, \ldots, h_k)$ we efficiently obtain $Z_{M,X_1,\ldots,X_k}(f_1, \ldots, f_n)$.

The Coppersmith Heuristic. Unfortunately, there is no *provable* guarantee that the variety of $\mathfrak{a}$ is zero-dimensional. In the multivariate setting, Coppersmith's method thus relies on the following (well-established) heuristic.

Heuristic 1 (Coppersmith Heuristic). *The polynomials obtained from Coppersmith's method generate an ideal of a zero-dimensional variety.*

While one can deliberately construct polynomials $f_1, \ldots, f_n$ and moduli M for which Heuristic 1 fails (see, e.g., [15, Section 12]), the heuristic holds for most instances arising in practice.

Nevertheless, we stress that it is important to verify the correctness of Heuristic 1 experimentally, since there are a few instances known for which the heuristic unexpectedly fails (see, e.g., the discussion on [48] in [34, Section 4]). We verify the correctness of Heuristic 1 for our algorithm for CI-HNP in Section 6.

Constructing $h_1, \ldots, h_k$. To construct the polynomials $h_1, \ldots, h_k$, Coppersmith's method requires as input a set of polynomials $\mathcal{F} \subset \mathbb{Z}[x_1, \ldots, x_k]$ satisfying certain technical constraints. (The h_i's are then computed as integer linear combinations of the elements of $\mathcal{F}$.) Construction of $\mathcal{F}$ is often difficult and usually done in an ad-hoc fashion. Furthermore, proving that a given set $\mathcal{F}$ satisfies the required technical constraints is often very tedious.

To overcome these issues, we introduce in the following Section 3.2 our novel and automated approach to Coppersmith's method, which allows us to greatly simplify construction of $\mathcal{F}$.

3.2 Coppersmith's Method Automated

The main idea behind our automated approach to Coppersmith's method is the following Definition 5. It allows us to abstract away all technicalities arising from lattice theory in Coppersmith's method and to replace them by purely combinatorial constraints.

Definition 5. *Let $\mathcal{M}$ be a finite set of monomials, and let $\prec$ be a monomial order on $\mathcal{M}$. A set of polynomials $\mathcal{F}$ is called* $(\mathcal{M}, \prec)$-suitable*, if:*

1. *Every $f \in \mathcal{F}$ is defined over $\mathcal{M}$.*
2. *For every monomial $\lambda \in \mathcal{M}$ there is a unique polynomial $f \in \mathcal{F}$ with leading monomial λ (with respect to $\prec$).*

If $\mathcal{F}$ is $(\mathcal{M}, \prec)$-suitable and $\lambda \in \mathcal{M}$, then we denote by $\mathcal{F}[\lambda]$ the unique polynomial $f \in \mathcal{F}$ with leading monomial λ.

We note that similar but less general constraints on $\mathcal{F}$ have first been used in [33, Lemma 4]. Definition 5 now allows us to compactly formulate Coppersmith's method as follows.

Theorem 2 (Coppersmith's Method). *Suppose we are given a modulus $M \in \mathbb{N}$, polynomials $f_1, \ldots, f_n \in \mathbb{Z}_M[x_1, \ldots, x_k]$ and bounds $0 \leq X_1, \ldots, X_k \leq$*

M, where $k = \mathcal{O}(1)$. Furthermore, suppose we are given an integer $m \in \mathbb{N}$, a set of monomials $\mathcal{M}$, a monomial order $\prec$ on $\mathcal{M}$, and an $(\mathcal{M}, \prec)$-suitable set of polynomials $\mathcal{F} \subseteq \mathbb{Z}_{M^m}[x_1, \dots, x_k]$ with

$$Z_{M,X_1,\dots,X_k}(f_1, \dots, f_n) \subseteq Z_{M^m,X_1,\dots,X_k}(\mathcal{F}). \tag{5}$$

If the conditions

$$\prod_{\lambda \in \mathcal{M}} |\mathsf{LC}(\mathcal{F}[\lambda])| \leq \frac{M^{(m-k)|\mathcal{M}|}}{\prod_{\lambda \in \mathcal{M}} \lambda(X_1, \dots, X_k)}, \tag{6}$$

$\log(M) \geq |\mathcal{M}| \geq m$ *and* $|\mathcal{M}| \geq k$ *hold, then we can compute all*

$$r \in Z_{M,X_1,\dots,X_k}(f_1, \dots, f_n)$$

in time polynomial in $\deg(\mathcal{F}) \cdot \log(M)$*, under Heuristic 1 for* $k > 1$.

A proof for Theorem 2 is given in Section 4. The algorithm behind Theorem 2 is given in Algorithm 1.

Algorithm 1: Coppersmith's Method.

Input: Integers $M, m \in \mathbb{N}$, polynomials $f_1, \dots, f_n \in \mathbb{Z}_M[x_1, \dots, x_k]$, bounds $0 \leq X_1, \dots, X_k \leq M$, set of monomials $\mathcal{M}$, monomial order $\prec$ on $\mathcal{M}$, and a $(\mathcal{M}, \prec)$-suitable set of polynomials $\mathcal{F} \subseteq \mathbb{Z}_{M^m}[x_1, \dots, x_k]$, satisfying the constraints of Theorem 2.

Output: All $r \in Z_{M,X_1,\dots,X_k}(f_1, \dots, f_n)$.

1 Construct an $|\mathcal{M}| \times |\mathcal{M}|$ basis matrix $\mathbf{B}$, whose columns are the coefficient vectors of the polynomials $\mathcal{F}[\lambda](X_1 x_1, \dots, X_k x_k)$, where $\lambda \in \mathcal{M}$.

2 LLL-reduce $\mathbf{B}$.

3 Interpret the first k column of the resulting matrix as coefficient vectors of polynomials $h_i(X_1 x_1, \dots, X_k x_k)$.

4 Compute the Gröbner basis of $\Big(h_1(x_1, \dots, x_k), \dots, h_k(x_1, \dots, x_k)\Big)$.

5 **return** all $r \in Z_{\mathbb{Z}}(h_1, \dots, h_k) \cap Z_{M,X_1,\dots,X_k}(f_1, \dots, f_n)$.

Given a modulus $M \in \mathbb{N}$, polynomials $f_1, \dots, f_n \in \mathbb{Z}_M[x_1, \dots, x_k]$ and bounds $X_1, \dots, X_k \in \mathbb{N}$, Theorem 2 now suggests the following simple approach for computing all small modular roots $r \in Z_{M,X_1,\dots,X_k}(f_1, \dots, f_n)$:

1. Pick a set of monomials $\mathcal{M}$ in $x_1, \dots, x_k$ with $|\mathcal{M}| \geq k$, a monomial order $\prec$ on $\mathcal{M}$, and an $m \in \mathbb{N}$, such that $\log(M) \geq |\mathcal{M}| \geq m$.
2. Pick an $(\mathcal{M}, \prec)$-suitable set of polynomials $\mathcal{F} \subseteq \mathbb{Z}_{M^m}[x_1, \dots, x_k]$, satisfying Equations (5) and (6).
3. Apply Theorem 2 / Algorithm 1 to compute all $r \in Z_{M,X_1,\dots,X_k}(f_1, \dots, f_n)$.

As we show below, choosing an *optimal* $\mathcal{F}$ can be automated entirely, once $\mathcal{M}$, m and $\prec$ are fixed. Furthermore, choosing m and $\prec$ is very easy. In our

new approach all difficulties of Coppersmith's method thus boil down the much simpler task of choosing $\mathcal{M}$.

Below, we also describe a very simple and automated strategy for choosing $\mathcal{M}$. While this strategy does not always yield optimal results, it still performs well in practice.

Choosing $\mathcal{F}$. Suppose we have already chosen $\mathcal{M}$, $\prec$ and m. The set $\mathcal{F}$ then has to satisfy the following three conditions:

1. It has to be $(\mathcal{M}, \prec)$-suitable,
2. it has to satisfy Equation (5),
3. it has to satisfy Equation (6).

Satisfying Equation (5) is easy: Like in all other Coppersmith-type results, we simply construct $\mathcal{F}$ using so-called *shift-polynomials*, i.e., polynomials of the form

$$p_{[j_1,\ldots,j_k,i_1,\ldots,i_n]} := x_1^{j_1} \cdot \ldots \cdot x_k^{j_k} \cdot f_1^{i_1} \cdot \ldots \cdot f_n^{i_n} \cdot M^{m-(i_1+\ldots+i_n)}, \tag{7}$$

for some appropriately chosen integers $j_1, \ldots, j_k, i_1, \ldots, i_n \in \mathbb{N}$, where $i_1 + \ldots + i_n \leq m$. Since for any $r \in Z_{M,X_1,\ldots,X_k}(f_1, \ldots, f_n)$ we have

$$f_1^{i_1}(r) \cdot \ldots \cdot f_n^{i_n}(r) \equiv 0 \mod M^{i_1+\ldots+i_n},$$

it then holds that

$$p_{[j_1,\ldots,j_k,i_1,\ldots,i_n]}(r) \equiv 0 \mod M^m.$$

The resulting $\mathcal{F} := \{p_{[j_1,\ldots,j_k,i_1,\ldots,i_n]}\}_{j_1,\ldots,j_k,i_1,\ldots,i_n}$ thus satisfies Equation (5).

For satisfying Equation (6), notice that the right hand side in Equation (6) does not depend on $\mathcal{F}$. For fixed $\mathcal{M}$, $\prec$, and m, Equation (6) thus simply requires that the product of (the absolute values of) the leading coefficients of the polynomials in $\mathcal{F}$ is smaller than some constant. Making the mild assumption that the f_i's are monic, it follows from Equation (4) that the leading coefficient of the shift-polynomial $p_{[j_1,\ldots,j_k,i_1,\ldots,i_n]}$ from Equation (7) is

$$\mathsf{LC}\left(p_{[j_1,\ldots,j_k,i_1,\ldots,i_n]}\right) = M^{m-(i_1+\ldots+i_n)}.$$

Hence, the larger the sum $i_1+\ldots+i_n$ gets, the smaller gets the leading coefficient of the corresponding shift-polynomial. To satisfy Equation (6), we thus have to take shift polynomials $p_{[j_1,\ldots,j_k,i_1,\ldots,i_n]}$ with as large $i_1 + \ldots + i_n$ as possible.

Finally, to ensure that $\mathcal{F}$ is $(\mathcal{M}, \prec)$-suitable, we have to include for every monomial $\lambda \in \mathcal{M}$ one shift-polynomial $p_{[j_1,\ldots,j_k,i_1,\ldots,i_n]}$ in $\mathcal{F}$, such that

1. the leading monomial of $p_{[j_1,\ldots,j_k,i_1,\ldots,i_n]}$ is λ, and
2. $p_{[j_1,\ldots,j_k,i_1,\ldots,i_n]}$ is defined over $\mathcal{M}$.

From Equation (3) it easily follows that the shift-polynomials, which satisfy these conditions, are exactly the polynomials of the form

$$f_{[\lambda,i_1,\ldots,i_n]} := \frac{\lambda}{\mathsf{LM}(f_1)^{i_1} \cdot \ldots \cdot \mathsf{LM}(f_n)^{i_n}} \cdot f_1^{i_1} \cdot \ldots \cdot f_n^{i_n} \cdot M^{m-(i_1+\ldots+i_n)}, \tag{8}$$

where

1. $\mathsf{LM}(f_1)^{i_1} \cdot \ldots \cdot \mathsf{LM}(f_n)^{i_n}$ divides λ, and
2. $f_{[\lambda,i_1,\ldots,i_n]}$ is defined over $\mathcal{M}$.

Hence, to construct an *optimal* set of shift-polynomials, we simply have to enumerate all such shift-polynomials $f_{[\lambda,i_1,\ldots,i_n]}$ and then include for every $\lambda \in \mathcal{M}$ one shift-polynomial in $\mathcal{F}$, that maximizes the sum $i_1 + \ldots + i_n$.

A formal description of this approach is given in Algorithm 2. The runtime of Algorithm 2 is $\mathcal{O}(|\mathcal{M}| \cdot m^n)$, which – for fixed n – is polynomial in our main parameter $\log(M)$, since by construction $m \leq |\mathcal{M}| \leq \log(M)$.

A somewhat optimized implementation of the algorithm is available in our GitHub repository. As we show in Section 6, our implementation is very efficient, even for cryptographically-sized instances.

Algorithm 2: Constructing an optimal set $\mathcal{F}$.

Input: Set of monomials $\mathcal{M}$, monomial order $\prec$ on $\mathcal{M}$, monic polynomials $f_1, \ldots, f_n$, and integer $m \in \mathbb{N}$.
Output: $(\mathcal{M}, \prec)$-suitable set of shift-polynomials $\mathcal{F}$, satisfying Equation (5), and minimizing the left hand side in Equation (6).

$\mathcal{F} := \emptyset$
for $\lambda \in \mathcal{M}$ **do**
  Enumerate all shift-polynomials $f_{[\lambda,i_1,\ldots,i_n]}$, as in Equation (8), such that $\mathsf{LM}(f)^{i_1} \cdot \ldots \cdot \mathsf{LM}(f)^{i_n}$ divides λ, and $f_{[\lambda,i_1,\ldots,i_n]}$ is defined over $\mathcal{M}$.
  Among all such $f_{[\lambda,i_1,\ldots,i_n]}$ pick one that maximizes $i_1 + \ldots + i_n$ and include it in $\mathcal{F}$.
end
return $\mathcal{F}$

Choosing $\prec$. The choice of $\prec$ is usually of secondary importance in Coppersmith's method, and simply choosing the lexicographic order $\prec_{\mathsf{lex}}$ will suffice in most applications. Indeed, if we were to restate all known Coppersmith-type results from the literature using the language of our new Theorem 2, then almost all results would use $\prec_{\mathsf{lex}}$ as monomial order.[3]

Choosing m and $\mathcal{M}$. Instead of choosing one fixed m and $\mathcal{M}$, we define an increasing sequence $\mathcal{M}_1 \subset \mathcal{M}_2 \subset \mathcal{M}_3 \subset \ldots$ of sets of monomials. While the Coppersmith-type literature strongly suggests that there is no *fully* automated strategy for choosing the $\mathcal{M}_i$'s, it appears that defining

$$\mathcal{M}_i := \left\{ \lambda \mid \lambda \text{ is a monomial of } f_1^{j_1} \cdot \ldots \cdot f_n^{j_n}, 0 \leq j_1, \ldots, j_n \leq i \right\} \tag{9}$$

$$m_i := i \cdot n, \tag{10}$$

[3] However, we note that some results, which deeply exploit the algebraic structure of the underlying polynomials $f_1, \ldots, f_n$ via *unravelled linearization* [24], e.g., [49,50,33], would require more involved monomial orders.

often provides a good starting point, which one then may further optimize by incorporating special properties of the underlying polynomials $f_1, \ldots, f_n$.

The condition from Equation (6), under which Coppersmith's method is successful, then typically translates to an inequality of the form

$$X_1^{\alpha_1} \cdot \ldots \cdot X_k^{\alpha_k} \leq M^{\delta - \varepsilon}, \tag{11}$$

for some constants $\alpha_1, \ldots, \alpha_k, \delta \geq 0$ and some $\varepsilon > 0$ that tends to 0 as $\mathcal{M}_i$ increases. (In other words, the larger we pick $\mathcal{M}_i$, the better Coppersmith's method performs.)

For the best possible result, we thus always pick the largest $\mathcal{M}_i$ that satisfies the condition $|\mathcal{M}_i| \leq \log(M)$ (which is imposed by Theorem 2). A typical Coppersmith-type result thus is a bound on the X_i's as in Equation (11), where the error term ε vanishes as $M \to \infty$.

Computing Asymptotic Bounds. Once we have chosen our sequence of sets $\mathcal{M}_i$, we can use Algorithm 2 to construct – for any fixed $\mathcal{M}_i$ and $m_i := i \cdot n$ – a corresponding optimal set of shift-polynomials $\mathcal{F}_i$. Given $\mathcal{F}_i$, $\mathcal{M}_i$ and m_i, we may then derive from Equation (6) a bound on $X_1, \ldots, X_k$, under which Coppersmith's method successfully recovers the desired small roots.

However, in practice one usually is not interested in the performance of Coppersmith's method for one fixed i, but rather in its asymptotic performance, i.e., usually it is desirable to obtain asymptotic bounds as in Equation (11). Luckily, Algorithm 2 also allows us to derive such asymptotic bounds via polynomial interpolation as follows:

It turns out that the terms in Equation (6) grow in practice as

$$M^{(m-k)|\mathcal{M}_i|} = M^{p_{\mathcal{M}}(m_i)}, \tag{12}$$

$$\prod_{\lambda \in \mathcal{M}_i} |\mathsf{LC}(\mathcal{F}_i[\lambda])| = M^{p_{\mathcal{F}}(m_i)} \tag{13}$$

$$\prod_{\lambda \in \mathcal{M}_i} \lambda(X_1, \ldots, X_k) = X_1^{p_1(m_i)} \cdot \ldots \cdot X_k^{p_k(m_i)}, \tag{14}$$

where $p_{\mathcal{M}}, p_{\mathcal{F}}, p_1, \ldots, p_k$ are polynomials of degree $k+1$. Based on this observation, we simply run Algorithm 2 on $\mathcal{M}_1, \ldots, \mathcal{M}_{k+2}$ to construct $k+2$ sets of shift-polynomials $\mathcal{F}$. Given $\mathcal{M}_1, \ldots, \mathcal{M}_{k+2}$ and the corresponding $\mathcal{F}_i$'s, we obtain the values of the polynomials $p_{\mathcal{F}}, p_{\mathcal{M}}, p_1, \ldots, p_k$ on $k+2$ different inputs. Using polynomial interpolation, we then easily construct $p_{\mathcal{M}}, p_{\mathcal{F}}, p_1, \ldots, p_k$.

Denoting the leading coefficients of the polynomials by $\ell_{\mathcal{M}}, \ell_{\mathcal{F}}, \ell_1, \ldots, \ell_k$, Equation (6) then translates to an asymptotic bound

$$X_1^{\ell_1} \cdot \ldots \cdot X_k^{\ell_k} \leq M^{\ell_{\mathcal{M}} - \ell_{\mathcal{F}} - \varepsilon}, \tag{15}$$

for some $\varepsilon > 0$ that vanishes as m increases, similar to Equation (11).

When defining $\mathcal{M}_i$ and m_i as in Equations (9) and (10), it is easy to see that exponents in Equations (12) and (14) indeed grow as polynomials in m_i.

However, *proving* that the same also holds for Equation (13) appears to be difficult. Therefore, we require the following heuristic assumption.

Heuristic 2. *Let* $f_1, \ldots, f_n \in \mathbb{Z}[x_1, \ldots, x_k]$*, let* $\prec$ *be a monomial order on* $x_1, \ldots, x_k$*, and define*

$$\mathcal{M}_i := \left\{ \lambda \mid \lambda \text{ is a monomial of } f_1^{j_1} \cdot \ldots \cdot f_n^{j_n}, 0 \leq j_1, \ldots, j_n \leq i \right\}$$
$$m_i := i \cdot n,$$

for $i \in \mathbb{N}$*. Then there exists a polynomial* $p(m)$ *of degree* $k+1$*, such that for any set* $\mathcal{F}_i$*, that is obtained from Algorithm 2 on input* $(\mathcal{M}_i, \prec, (f_1, \ldots, f_n), m_i)$*, it holds that*

$$\prod_{\lambda \in \mathcal{M}_i} |\mathsf{LC}(\mathcal{F}_i[\lambda])| = M^{p(m_i)}.$$

In practice, Heuristic 2 always seems to hold. It is in an interesting open problem to further explore this behavior of Algorithm 2.

We note that in order to increase confidence in Heuristic 2 for any given set of polynomials $\{f_1, \ldots, f_n\}$, one may construct significantly more than $k+2$ sets $\mathcal{M}_i$ with corresponding sets of shift-polynomials $\mathcal{F}_i$. If the polynomial interpolation then still yields a polynomial of degree $k+1$, this serves as a very strong indication of the correctness of Heuristic 2.

If one still wishes to rigorously prove asymptotic bounds, i.e. without Heuristic 2, then one can proceed as follows: We run Algorithm 2 on $\mathcal{M}_1, \ldots, \mathcal{M}_{k+2}$, but instead of using polynomial interpolation, we (manually) look for patterns in the algorithms output, i.e., we look for patterns in the resulting sets of shift-polynomials $\mathcal{F}_i$. From these patterns, we then derive formulas that describe for any given $\mathcal{M}_i$ the corresponding set $\mathcal{F}_i$. Finally, these formulas allow us to derive an asymptotic bound as in Equation (15).[4]

Clearly, this approach is significantly less automated than our polynomial interpolation approach based on Heuristic 2. However, due to the use of Algorithm 2 it is arguably still much simpler than most previous approaches to Coppersmith's method.

3.3 Applications of our Automated Approach

Let us now use our automated approach to Coppersmith's method to derive two new Coppersmith-type bounds, which we use in the subsequent Section 5 to prove our results for CI-HNP.

Theorem 3. *Suppose we are given a modulus* $M \in \mathbb{N}$*, polynomials*

$$f(x, y, z) := xy + f_1 x + f_2 y + f_3,$$
$$g(x, y, z) := yz + g_1 y + g_2 z + g_3,$$
$$h(x, y, z) := xz + h_1 x + h_2 z + h_3,$$

[4] We note that this approach is similar to the integer programming approach recently introduced by May, Nowakowski and Sarkar [33, Remark 1].

for some constants $f_i, g_i, h_i \in \mathbb{Z}$*, bounds* $X, Y, Z \in \mathbb{N}$*, and an arbitrarily small constant* $\varepsilon > 0$*. If* M *is sufficiently large, and*

$$XYZ < M^{11/8-\varepsilon},$$

then we can compute all $r \in Z_{M,X,Y,Z}(f, g, h)$ *in time polynomial in* $\log(M)$*, under Heuristics 1 and 2.*

Proof. Following our strategy from Section 3.2, we choose a parameter $i \in \mathbb{N}$, define

$$\begin{aligned}\mathcal{M}_i &:= \left\{\lambda \mid \lambda \text{ is a monomial of } f^{j_1} g^{j_2} h^{j_3}, 0 \le j_1, j_2, j_3 \le i\right\}\\ m_i &= i \cdot 3,\end{aligned}$$

and equip the elements in $\mathcal{M}_i$ with the lexicographic monomial order $\prec_{\mathsf{lex}}$ on x, y, z. Note that the constraints $|\mathcal{M}_i| \ge m_i$ and $|\mathcal{M}_i| \ge 3$ from Theorem 2 are trivially satisfied. For sufficiently large M, the additional constraint $\log(M) \ge |\mathcal{M}_i|$ is also satisfied.

It is easy to see that

$$\begin{aligned}M^{(m-3)|\mathcal{M}_i|} &= M^{p_{\mathcal{M}}(m_i)},\\ \prod_{\lambda \in \mathcal{M}_i} \lambda(X, Y, Z) &= X^{p_X(m_i)} \cdot Y^{p_Y(m_i)} \cdot Z^{p_Z(m_i)},\end{aligned}$$

for some polynomials $p_{\mathcal{M}}, p_X, p_Y, p_Z$ of degree 4. Under Heuristic 2, we also have

$$\prod_{\lambda \in \mathcal{M}_i} |\mathsf{LC}(\mathcal{F}_i[\lambda])| = M^{p_{\mathcal{F}}(m_i)}$$

for some polynomial $p_{\mathcal{F}}$ of degree 4, where $\mathcal{F}_i$ denotes the output of Algorithm 2 on input $(\mathcal{M}_i, \prec_{\mathsf{lex}}, (f, g, h), m_i)$.

We run Algorithm 2 for $i = 1, \ldots, 5$. From the output of the algorithm, we obtain the following values:

m_i	$p_{\mathcal{M}}(m_i)$	$p_{\mathcal{F}}(m_i)$	$p_X(m_i)$	$p_Y(m_i)$	$p_Z(m_i)$
3	0	50	27	27	27
6	375	439	250	250	250
9	2058	1767	1029	1029	1029
12	6561	4946	2916	2916	2916
15	15972	11200	6655	6655	6655

Using polynomial interpolation, we obtain

$$\begin{aligned}p_{\mathcal{M}}(m_i) &= \frac{8}{27} m_i^4 + o(m_i^4),\\ p_{\mathcal{F}}(m_i) &= \frac{13}{81} m_i^4 + o(m_i^4),\\ p_X(m) = p_Y(m_i) &= p_Z(m_i) = \frac{8}{81} m_i^4 + o(m_i^4).\end{aligned}$$

Hence, the condition from Equation (6) becomes

$$X^{8/81}Y^{8/81}Z^{8/81} < M^{8/27-13/81-\varepsilon} = M^{11/81-\varepsilon}$$

for some $\varepsilon > 0$ that vanishes as m (or equivalently M) increases. Taking the $\frac{8}{81}$-th root and replacing ε by $\frac{81}{8}\varepsilon$ in the above inequality, we obtain

$$XYZ < M^{11/8-\varepsilon},$$

as required. □

In the full version of the paper, we show that Theorem 3 remains correct even when removing Heuristic 2 from the theorem. (However removing the heuristic comes at the cost of a significantly more complicated proof and manual effort.) We see this as a strong indication of the correctness of Heuristic 2.

Theorem 4. *Suppose we are given a modulus $M \in \mathbb{N}$, polynomials*

$$f(x,y,z) := x^2 + f_1xy^2 + f_2xy + f_3x + f_4y^2 + f_5y + f_6,$$
$$g(x,y,z) := z^2 + g_1x^2z + g_2xz + g_3z + g_4x^2 + g_5x + g_6,$$

for some bounds $f_i, g_i \in \mathbb{Z}$, bounds $X, Y, Z \in \mathbb{N}$, and an arbitrarily small constant $\varepsilon > 0$. If M is sufficiently large, and

$$XYZ < M^{30/41-\varepsilon},$$

then we can compute all $r \in Z_{M,X,Y,Z}(f,g,h)$ in time polynomial in $\log(M)$, under Heuristics 1 and 2.

The proof of Theorem 4 is analogous to that of Theorem 3 and therefore omitted. A rigorous but involved proof that does not require Heuristic 2 is given in the full version of the paper.

The ε-term in Theorems 3 and 4. Previous works on Coppersmith's method often (implicitly) assume that one can easily eliminate the ε-term in Coppersmith-type bounds. However, as we discuss in the full version of the paper, when dealing with systems of *multivariate* equations (as in Theorems 3 and 4), the ε-term is inherent and eliminating it requires sub-exponential (but super-polynomial) runtime.

4 Proof for Theorem 2

The main idea behind Coppersmith's method is the following simple Lemma 2. Intuitively, it states that small modular roots of a polynomial h with small coefficients are actually integer roots of h.

Lemma 2 (Håstad [23], Howgrave-Graham [25]). *Let $h(x_1, \ldots, x_k)$ be a polynomial with at most d monomials, and let $M^m, X_1, \ldots, X_k \in \mathbb{N}$. Suppose h has a root $r = (r_1, \ldots, r_k)$ modulo M^m, satisfying $|r_i| \leq X_i$ for every $i = 1, \ldots, k$. If*

$$\|h(X_1 x_1, \ldots, X_k x_k)\| < \frac{M^m}{\sqrt{d}},$$

then $h(r_1, \ldots, r_k) = 0$ holds over the integers.

As discussed in Section 3.1, given a set of polynomials $\mathcal{F}$, Coppersmith's method efficiently computes all small modular roots $r \in Z_{M,X_1,\ldots,X_k}(\mathcal{F})$ by constructing a set of k-polynomials $\{h_1, \ldots, h_k\}$ such that

$$Z_{M,X_1,\ldots,X_k}(\mathcal{F}) \subseteq Z_{\mathbb{Z}}(h_1, \ldots, h_k).$$

To this end, Coppersmith's method uses LLL lattice reduction to construct the h_i's as small-norm integer linear combinations of the $f \in \mathcal{F}$. By Lemma 2 every r then is an integer root of the h_i's, as required.

Using this observation we now prove Theorem 2, which for the sake of readability we recall below.

Theorem 2 (Coppersmith's Method). *Suppose we are given a modulus $M \in \mathbb{N}$, polynomials $f_1, \ldots, f_n \in \mathbb{Z}_M[x_1, \ldots, x_k]$ and bounds $0 \leq X_1, \ldots, X_k \leq M$, where $k = \mathcal{O}(1)$. Furthermore, suppose we are given an integer $m \in \mathbb{N}$, a set of monomials $\mathcal{M}$, a monomial order $\prec$ on $\mathcal{M}$, and an $(\mathcal{M}, \prec)$-suitable set of polynomials $\mathcal{F} \subseteq \mathbb{Z}_{M^m}[x_1, \ldots, x_k]$ with*

$$Z_{M,X_1,\ldots,X_k}(f_1, \ldots, f_n) \subseteq Z_{M^m,X_1,\ldots,X_k}(\mathcal{F}). \tag{5}$$

If the conditions

$$\prod_{\lambda \in \mathcal{M}} |\mathsf{LC}(\mathcal{F}[\lambda])| \leq \frac{M^{(m-k)|\mathcal{M}|}}{\prod_{\lambda \in \mathcal{M}} \lambda(X_1, \ldots, X_k)}, \tag{6}$$

$\log(M) \geq |\mathcal{M}| \geq m$ and $|\mathcal{M}| \geq k$ hold, then we can compute all

$$r \in Z_{M,X_1,\ldots,X_k}(f_1, \ldots, f_n)$$

in time polynomial in $\deg(\mathcal{F}) \cdot \log(M)$, under Heuristic 1 for $k > 1$.

Proof. For every $i = 1, \ldots, |\mathcal{M}|$, let λ_i denote the i-th smallest monomial in $\mathcal{M}$ (with respect to $\prec$). For every $\lambda \in \mathcal{M}$, we denote by $\mathbf{f}_\lambda \in \mathbb{Z}^{|\mathcal{M}|}$ the coefficient vector of $\mathcal{F}[\lambda](X_1 x_1, \ldots, X_k x_k)$, where the i-th coordinate of $\mathbf{f}_\lambda$ is the coefficient of λ_i in $\mathcal{F}[\lambda](X_1 x_1, \ldots, X_k x_k)$.

We construct an $|\mathcal{M}| \times |\mathcal{M}|$ lattice basis matrix $\mathbf{B}$, where the i-th column of $\mathbf{B}$ is the vector $\mathbf{f}_{\lambda_i}$. Since λ_i is the leading monomial of $\mathcal{F}_{\lambda_i}$, the i-th entry of $\mathbf{f}_{\lambda_i}$ equals $\mathsf{LC}(\mathcal{F}_{\lambda_i}) \cdot \lambda_i(X_1, \ldots, X_k)$. Further, for every $j > i$ the j-th entry of $\mathbf{f}_{\lambda_i}$ equals 0, since $\lambda_i \prec \lambda_j$. Hence, $\mathbf{B}$ is a triangular matrix with determinant

$$\det \mathbf{B} = \prod_{\lambda \in \mathcal{M}} \mathsf{LC}(\mathcal{F}[\lambda]) \cdot \lambda(X_1, \ldots, X_k).$$

Together with Equation (6) this implies

$$\det \mathcal{L}(\mathbf{B}) \leq M^{(m-k)|\mathcal{M}|}. \tag{16}$$

We compute an LLL-reduced basis $\mathbf{B}_{\mathsf{LLL}} = (\mathbf{b}_1, \ldots, \mathbf{b}_{|\mathcal{M}|})$ of $\mathcal{L}(\mathbf{B})$. From $\log(M) \geq |\mathcal{M}| \geq m$, Lemma 1 and Equation (16) it follows that the first k columns $\mathbf{b}_1, \ldots, \mathbf{b}_k$ of $\mathbf{B}_{\mathsf{LLL}}$ have norm

$$\|\mathbf{b}_1\|, \ldots, \|\mathbf{b}_k\| < \frac{M^m}{\sqrt{|\mathcal{M}|}}. \tag{17}$$

Notice, since $|\mathcal{M}| \geq k$, the matrix $\mathbf{B}_{\mathsf{LLL}}$ indeed has at least k columns.

By definition of $\mathbf{B}$, every vector $\mathbf{b}_i$ from the LLL-reduced basis is the coefficient vector of some polynomial $h_i(X_1 x_1, \ldots, X_k x_k)$, such that

$$h_i(x_1, \ldots, x_k) = \sum_{\lambda \in \mathcal{M}} \alpha_{i,\lambda} \mathcal{F}[\lambda](x_1, \ldots, x_k), \tag{18}$$

for some $\alpha_{i,\lambda} \in \mathbb{Z}$. Let $r \in Z_{M,X_1,\ldots,X_k}(f_1, \ldots, f_n)$. Since

$$Z_{M,X_1,\ldots,X_k}(f_1, \ldots, f_n) \subseteq Z_{M^m,X_1,\ldots,X_k}(\mathcal{F}),$$

it follows from Equation (18) that

$$h_i(r) \equiv \sum_{\lambda \in \mathcal{M}} \alpha_{i,\lambda} \mathcal{F}[\lambda](r) \equiv 0 \mod M^m.$$

Together with Equation (17) and Lemma 2, this implies that r is a root of $h_1, \ldots, h_k$ have r over the integers. Hence,

$$Z_{M,X_1,\ldots,X_k}(f_1, \ldots, f_n) \subseteq Z_{\mathbb{Z}}(h_1, \ldots, h_k).$$

Since the entries of $\mathbf{B}$ are upper bounded by polynomials of degree at most $d := \deg(\mathcal{F})$ in M^m, the runtime of LLL to compute $\mathbf{B}_{\mathsf{LLL}}$ is polynomial in $d \cdot m \cdot \log(M) \leq d \log(M)^2$ and $|\mathcal{M}| \leq \log(M)$. Hence, we can compute $h_1, \ldots, h_k$ in time polynomial in $d \cdot \log(M)$, as required.

Finally, if $k = 1$, we efficiently obtain all $r \in Z_{M,X_1,\ldots,X_k}(f_1, \ldots, f_n)$, by computing all integer roots of h_1 and then outputting only those that lie in $Z_{M,X_1,\ldots,X_k}(f_1, \ldots, f_n)$. If $k > 1$, we efficiently obtain all such r's under Heuristic 1 from the Gröbner basis of $\mathfrak{a} := (h_1, \ldots, h_k)$, which we can compute in polynomial time, since $k = \mathcal{O}(1)$. □

5 The Commutative Isogeny Hidden Number Problem

In this section we use the results developed in Section 3 to solve the Commutative Isogeny Hidden Number Problem. We assume that an elliptic curve is always represented by its corresponding Montgomery coefficient. That is, oracles or

algorithms that take as input an elliptic curve always expect the Montgomery coefficient of said curve. The same principle applies to the output of such an oracle or algorithm. This means that we mainly work with the group action from Equation (1) and Equation (2), respectively.

We now define the main computational problem, which applies to both CSIDH and CSURF where we (implicitly) set the prime p, the order $\mathcal{O}$ and the starting curve M accordingly.

Definition 6 (Commutative Isogeny Hidden Number Problem ($\mathsf{CI\text{-}HNP}_k$)). *Let p be an n-bit prime and let $k < n$ be a positive integer. Further let $[\mathfrak{a}], [\mathfrak{b}] \xleftarrow{\$} cl(\mathcal{O})$. Assume that there exists an oracle $\mathcal{O}_{\mathsf{MSB}_k}$ that on input two Montgomery coefficients $M_0, M_1 \in \mathcal{M}_p(\mathcal{O})$ computes*

$$\mathcal{O}_{\mathsf{MSB}_k}(M_0, M_1) \coloneqq \mathsf{MSB}_k(\mathsf{CDH}(M_0, M_1)).$$

Given the tuple $(\mathfrak{a} \star M, \mathfrak{b} \star M)$ and access to $\mathcal{O}_{\mathsf{MSB}_k}$, the task is to recover $\mathfrak{ab} \star M$.

Remark 3. Because we can write

$$M_{\mathfrak{ab}} \coloneqq \mathfrak{ab} \star M = \mathcal{O}_{\mathsf{MSB}_k}(\mathfrak{a} \star M, \mathfrak{b} \star M) \cdot 2^{n-k} + m_{\mathfrak{ab}} \tag{19}$$

for some $m_{\mathfrak{ab}} < 2^{n-k}$ we focus on recovering $m_{\mathfrak{ab}}$ from now on.

In the next sections we give an algorithm $\mathcal{A}$ that solves $\mathsf{CI\text{-}HNP}_k$ for both CSIDH and CSURF. Like many algorithms that solve a flavour of the hidden number problem, $\mathcal{A}$ proceeds in two stages:

1. Query the oracle $\mathcal{O}_{\mathsf{MSB}_k}$ on specific input, obtaining a set of bivariate polynomial equations that have a common small root in $m_{\mathfrak{ab}}$.
2. Use Coppersmith's method to solve the system of equations, yielding the common root $m_{\mathfrak{ab}}$.

We remark that the second stage is the same for both CSIDH and CSURF. Furthermore, for the results in the following sections it is actually sufficient to have a *static* oracle. That is, an oracle where one of the inputs to the oracle is fixed, i.e.

$$\mathcal{O}_{\mathsf{MSB}_k}(M') \coloneqq \mathsf{MSB}_k(\mathsf{CDH}(M', \mathfrak{b} \star M))$$

with $\mathfrak{b}$ as in Definition 6.

5.1 Solving CI-HNP for CSIDH

Let $p \equiv 3 \mod 8$ be a prime and $\mathcal{O} = \mathbb{Z}[\pi]$. Our goal is to find polynomial relations between neighboring Montgomery curves similar to what the modular polynomial provides for j-invariants. A variant of Vélu's formula dedicated to Montgomery curves provides a good starting point.

Theorem 5 (Proposition 1 in [39]). *Let $E_A : y^2 = x^3 + Ax^2 + x$ with $A^2 \neq 4$ be a Montgomery curve defined over $\mathbb{F}_p$ and let $G \subset E_A(\overline{\mathbb{F}}_p)$ be a finite subgroup such that $(0,0) \notin G$. Further define φ to be a separable isogeny with $\ker \varphi = G$. Then there exists a Montgomery curve $E_B : y^2 = x^3 + Bx^2 + x$ such that, up to isomorphism, $\varphi : E_A \to E_B$ and*

$$B = \tau(A - 3\sigma), \qquad \text{where } \tau = \prod_{P \in G \setminus \{\infty\}} x_P, \qquad \sigma = \sum_{P \in G \setminus \{\infty\}} \left(x_P - \frac{1}{x_P} \right).$$

By expanding the equation for B we immediately get a polynomial that relates two isogenous Montgomery coefficients to each other. Evidently, Theorem 5 can handle isogenies of almost arbitrary degree d and therefore could be used to derive polynomials describing the neighborhood of any two d-isogenous curves. To keep the total degree of the polynomial low, however, it is beneficial to look at isogenies of small degree. It is therefore natural to consider 3-isogenies as they have the smallest kernel amongst those isogenies admissible by the CSIDH group action. Moreover, removing the unwanted variables $\{x_P\}_{P \in G \setminus \{\infty\}}$ can be done via the 3-*division polynomial* (for a precise definition see [9]) and a resultant computation.

In the case of 3-isogenies this approach yields polynomial relations of total degree 6. However, we can improve on this by instead considering *4-isogenies*. In fact, the ideal (4) splits in $\mathbb{Z}[\pi]$ as $\mathfrak{l}\bar{\mathfrak{l}} = \langle 4, \pi - 1 \rangle \langle 4, \pi + 1 \rangle$. Most notably, the ideal $\mathfrak{l}$ has order 3 which is a direct result of using the class group of the non-maximal order $\mathbb{Z}[\pi]$ [11,38]. Hence the following formulas are only applicable to the CSIDH setting.

Proposition 1 (Theorem 7 in [38]). *Let $A \in \mathcal{M}_p(\mathbb{Z}[\pi])$ be the Montgomery coefficient of the curve $E_A \in \mathcal{E}\ell\ell_p(\mathbb{Z}[\pi])$. The two 4-isogenous curves of E_A in $\mathcal{E}\ell\ell_p(\mathbb{Z}[\pi])$ are*

$$E_B : y^2 = x^3 + Bx^2 + x, \qquad \text{where } B = 2\frac{A-6}{A+2}$$

and

$$E_C : y^2 = x^3 + Cx^2 + x, \qquad \text{where } C = 2\frac{A+6}{2-A}.$$

It is immediately evident that due to their simple form, the 4-isogeny formulas result in polynomial relations of degree only 2.

Corollary 1. *The Montgomery coefficients A, B and C from Proposition 1 satisfy the relations*

$$2A - AB - 2B - 12 = 0,$$
$$2C - AC - 2A - 12 = 0 \text{ and}$$
$$2B - BC - 2C - 12 = 0.$$

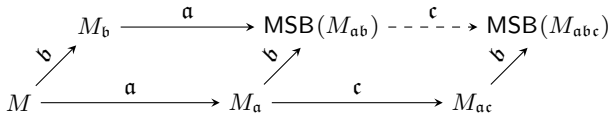

Fig. 3. Visualization of the general strategy where $\mathfrak{c}$ is an ideal of small norm (i.e. an isogeny of small degree). Here, $\mathcal{O}_{\mathsf{MSB}_k}$ allows us to compute the most significant bits of $M_{\mathfrak{ab}}$ and $M_{\mathfrak{abc}}$, which are connected by the same ideal $\mathfrak{c}$.

Proof. This is a simple restatement of Proposition 1 where the third formula is derived from the first two by taking the resultant with respect to A. □

Remarkably, due to the small order of $\mathfrak{l}$ we get *three* relations between A and its 4-isogenous neighbors B and C that all have the same total degree. This is particularly interesting as we can usually only hope to craft two polynomial relations of the same degree between A and its d-isogenous neighbors. The reason for this is that the two neighboring curves B and C are in general d^2-isogenous, resulting in a third polynomial relation of larger degree. As it turns out, Coppersmith's method strongly benefits from having a third relation of the same total degree, which in turn allows us to solve $\mathsf{CI\text{-}HNP}_k$ for a smaller value k.

Theorem 6. *Let $p \equiv 3 \mod 8$ be a n-bit prime, and let $\varepsilon > 0$ be an arbitrarily small constant. There exists a* PPT *algorithm $\mathcal{A}$ that solves $\mathsf{CI\text{-}HNP}_k$ in the CSIDH setting for $k = (\frac{13}{24} + \varepsilon)n$ under Heuristic 1.*

Proof. Let $(M_{\mathfrak{a}}, M_{\mathfrak{b}})$ be an instance of $\mathsf{CI\text{-}HNP}_k$. The algorithm $\mathcal{A}$ proceeds as follows: First, it uses Proposition 1 to compute the 4-isogenous neighbors of $M_{\mathfrak{a}}$, which we denote by $M_{\mathfrak{ac}}$ and $M_{\mathfrak{ad}}$ respectively. It then submits the queries $\mathcal{O}_{\mathsf{MSB}_k}(M_{\mathfrak{a}}, M_{\mathfrak{b}})$ and $\mathcal{O}_{\mathsf{MSB}_k}(M_{\mathfrak{ac}}, M_{\mathfrak{b}})$, which yield the most significant bits of $M_{\mathfrak{ab}} = \mathsf{CDH}(M_{\mathfrak{a}}, M_{\mathfrak{b}})$ and $M_{\mathfrak{abc}} = \mathsf{CDH}(M_{\mathfrak{ac}}, M_{\mathfrak{b}})$ (see Figure 3). Since the group action is commutative we have that $M_{\mathfrak{ab}}$ and $M_{\mathfrak{abc}}$ are 4-isogenous as well, thus satisfying the first equation in Corollary 1. The same process is repeated for $M_{\mathfrak{ad}}$, yielding the most significant bits of $M_{\mathfrak{abd}}$. Finally, by using Equation (19) we can rewrite the resulting equations in terms of the least significant bits $m_{\mathfrak{ab}}$, $m_{\mathfrak{abc}}$ and $m_{\mathfrak{abd}}$, which are now small roots of size $p^{11/24}$ of the respective polynomials. $\mathcal{A}$ then finds these small roots via Theorem 3. □

5.2 Solving CI-HNP for CSURF

Let $p \equiv 7 \mod 8$ be a prime and $\mathcal{O} = \mathbb{Z}[(1 + \pi)/2]$. We use a very similar strategy compared to Section 5.1 to craft the polynomials. Unfortunately, we cannot use the same trick involving 4-isogenies from Proposition 1 as they are specific to the CSIDH setting. Instead, we can consider 2-isogenies since we have the ideals $\mathfrak{l}_0$ and $\bar{\mathfrak{l}}_0$ available. The resulting formulas still have small degree but cannot quite compete with the formulas for 4-isogenies in CSIDH. In particular,

the ideal $\mathfrak{l}_0$ has very large order, meaning that we only get two relations between a curve and its 2-isogenous neighbors.

Recall that the point $P_{\rightarrow} = (0, 0)$ has order 2 and corresponds to the ideal $\mathfrak{l}_0$. In order to compute the image curve of the vertical isogeny with kernel $\langle P_{\rightarrow} \rangle$ we use the following formula from [18, Equation (18)].

Proposition 2. *Let $A \in \mathcal{M}_p(\mathbb{Z}[(1+\pi)/2])$ be the Montgomery coefficient of the curve E_A and let $P_{\rightarrow} = (0, 0)$. The curve $E_A/\langle P_{\rightarrow} \rangle$ is isomorphic to a Montgomery curve E_B that can be written as*

$$E_B : y^2 = x^3 + Bx^2 + x, \qquad \text{where } B = \frac{A+6}{2\sqrt{A+2}}.$$

Note that $A + 2$ is a square by definition. Squaring both sides and rearranging terms yields a bivariate polynomial of total degree 3.

Corollary 2. *Let the notation be as in Proposition 2. The Montgomery coefficients A and B satisfy*

$$A^2 + 12A - 4B^2A - 8B^2 + 36 = 0.$$

The formula above only applies to the vertical isogeny generated by the point $P_{\rightarrow}$. However, one can treat the other vertical isogeny generated by $P_{\leftarrow} \neq (0, 0)$ similarly by observing that if $E' = E/\langle P_{\leftarrow} \rangle$, then $E \cong E'/\langle P_{\rightarrow} \rangle$. We thus get almost the same formula as in Corollary 2 where the only difference is that the coefficient A now takes the role of the image curve.

Corollary 3. *Let $A \in \mathcal{M}_p(\mathbb{Z}[(1+\pi)/2])$ be the Montgomery coefficient of E_A and let E_C be the Montgomery curve isomorphic to $E_A/\langle P_{\leftarrow} \rangle$. Then the Montgomery coefficients A and C satisfy*

$$C^2 + 12C - 4A^2C - 8A^2 + 36 = 0.$$

Observe that in Corollary 3 the monomials involving the Montgomery coefficient A are quite different compared to Corollary 2. This is in stark contrast to the CSIDH setting (in particular Corollary 1) where in the first two equations the monomials involving A are almost identical (up to sign). This "asymmetry" in the polynomials is undesirable for Coppersmith's method. In combination with the fact that we only have two relations instead of three, we have to increase the value k significantly in order to solve $\mathsf{CI\text{-}HNP}_k$ for CSURF.

Theorem 7. *Let $p \equiv 7 \mod 8$ be a n-bit prime, and let $\varepsilon > 0$ be an arbitrarily small constant. There exists a PPT algorithm $\mathcal{A}$ that solves $\mathsf{CI\text{-}HNP}_k$ in the CSURF setting for $k = (\frac{31}{41} + \varepsilon)n$ under Heuristic 1.*

Proof. The algorithm $\mathcal{A}$ proceeds like in the previous section. Given an instance $(M_{\mathfrak{a}}, M_{\mathfrak{b}})$ of $\mathsf{CI\text{-}HNP}_k$, $\mathcal{A}$ first computes the 2-isogenous coefficients $M_{\mathfrak{ac}}$ and $M_{\mathfrak{ad}}$ by quotienting out $\langle P_{\rightarrow} \rangle$ and $\langle P_{\leftarrow} \rangle$ on $M_{\mathfrak{a}}$, respectively. It then submits the

Fig. 4. Overview of the reduction from simulating the oracle $\mathcal{O}_{\mathsf{MSB}_k}$ to DLOG. Dashed lines denote quantum reductions.

oracle queries $\mathcal{O}_{\mathsf{MSB}_k}(M_{\mathfrak{a}}, M_{\mathfrak{b}})$ and $\mathcal{O}_{\mathsf{MSB}_k}(M_{\mathfrak{ac}}, M_{\mathfrak{b}})$, which yield the most significant bits of the coefficients $M_{\mathfrak{ab}} = \mathsf{CDH}(M_{\mathfrak{a}}, M_{\mathfrak{b}})$ and $M_{\mathfrak{abc}} = \mathsf{CDH}(M_{\mathfrak{ac}}, M_{\mathfrak{b}})$. Lastly it uses Equation (19) to express the equation from Corollary 2 in terms of $m_{\mathfrak{ab}}$ and $m_{\mathfrak{abc}}$, where $m_{\mathfrak{ab}}$ and $m_{\mathfrak{abc}}$ are now small roots of size $p^{10/41}$ of the corresponding polynomial. The same process is repeated with the curve $M_{\mathfrak{ad}}$ and Corollary 3. The small root $m_{\mathfrak{ab}}$ is then found by Coppersmith's method and the bound for k follows from Theorem 4. Note that the monomials in Theorem 4 differ slightly from those appearing in Corollary 2 and Corollary 3 due to the substitution mentioned in Equation (19). □

5.3 Hardness of Simulating $\mathcal{O}_{\mathsf{MSB}_k}$

The results from the previous sections can be directly used to analyze the hardness of simulating the oracle $\mathcal{O}_{\mathsf{MSB}_k}$. More concretely, simulating $\mathcal{O}_{\mathsf{MSB}_k}$ is quantumly as hard as solving DLOG due to the equivalence of CDH and DLOG in the CSIDH/CSURF setting. For simplicity we state the following result only for CSIDH, the statement and proof for CSURF is completely analogous.

Corollary 4. *Let $p \equiv 3 \mod 4$ be an n-bit prime, $\mathcal{O} = \mathbb{Z}[\pi]$ and $k = (\frac{13}{24} + \varepsilon)n$ for some arbitrary small constant $\epsilon > 0$. Assume that there exists an efficient (possibly quantum) algorithm $\mathcal{A}$ with*

$$\Pr[\mathcal{A}(\mathfrak{a} \star M, \mathfrak{b} \star M) = \mathcal{O}_{\mathsf{MSB}_k}(\mathfrak{a} \star M, \mathfrak{b} \star M)] = 1$$

where $[\mathfrak{a}], [\mathfrak{b}] \xleftarrow{\$} cl(\mathcal{O})$. Then there exists an efficient quantum algorithm $\mathcal{B}$ solving DLOG *in the CSIDH setting under Heuristic 1.*

Proof. The reduction is straightforward and depicted in Figure 4. In a first step, we use our algorithm developed in Theorem 6 to transform the algorithm $\mathcal{A}$ into an algorithm $\mathcal{A}'$ solving CDH under Heuristic 1. In a second step we can simply use $\mathcal{A}'$ (which still has success probability 1) together with the techniques developed by [52] to construct the algorithm $\mathcal{B}$ solving DLOG. □

We currently require $\mathcal{A}$ to simulate $\mathcal{O}_{\mathsf{MSB}_k}$ perfectly. This is a direct consequence of the fact that there is no obvious way to re-randomize the inputs to the oracle $\mathcal{O}_{\mathsf{MSB}_k}$ such that we still get meaningful information about the neighboring curves of $\mathfrak{ab} \star M$.

6 Experimental Results

We implemented our new automated variant of Coppersmith's method from Theorem 2 in SageMath and used it to run our algorithms from Theorems 6 and 7 in practice.

CSIDH and CSURF results. We ran our algorithms from Theorems 6 and 7 using SageMath 9.7 on an AMD EPYC 7763 processor with 128 physical and 256 logical cores. As Table 1 shows, our algorithms perform well in practice and we come close to our asymptotic bounds of $k = \frac{13}{24}n \approx 0.542n$ and $k = \frac{31}{41}n = 0.756n$ in a matter of minutes.

In every experiment Heuristic 1 was valid, i.e., we were always able to extract the unknowns from the Gröbner basis. This confirms the correctness of our heuristic algorithms.

Implementation Details. For constructing the set of shift-polynomials $\mathcal{F}$, we used in our experiments a slightly optimized implementation of Algorithm 2. Instead of simply enumerating *all* possible shift-polynomials in Step 3 of the algorithm, our implementation iterates over a carefully crafted tree of shift-polynomials. The tree is constructed only implicitly, and our implementation automatically detects (and ignores) some branches that are not worth visiting. This results in a significant speed-up in practice.

The LLL lattice reduction step was performed using the recently published `flatter` algorithm [42], which significantly outperforms SageMath's native implementation of LLL (which internally calls `FPLLL`).

For the Gröbner basis computation, we used SageMath's native Gröbner basis algorithm (which internally calls `Singular`) to compute Gröbner bases over small finite fields $\mathbb{F}_2, \mathbb{F}_3, \mathbb{F}_5, \mathbb{F}_7 \ldots$, and then recovered the desired roots via Chinese remaindering.

7 Conclusion

In this work we analyzed the Commutative Isogeny Hidden Number Problem and solved it for $k = \frac{13}{24}n$ (CSIDH) and $k = \frac{31}{41}n$ (CSURF) by using a new and automated variant of Coppersmith's method. Since the recovery rate for CSURF is much worse compared to CSIDH, we conclude that in the context of side-channel attacks, CSURF offers more resilience against exposing the most significant bits of the shared key. Even more generally it seems to be advisable that the class group $cl(\mathcal{O})$ does not contain a small order subgroup, which is in line with previous observations [11,38].

Furthermore, we gave a purely combinatorial restatement of Coppersmith's method that allows for near complete automation. In particular, we identified a single step in Coppersmith's method that, when optimized, yields provably optimal results. We implemented our variant of Coppersmith's method in SageMath

n	k (known bits)	m	$\lvert\mathcal{M}\rvert$ (lattice dim.)	Runtime		
				$\mathcal{F}$	LLL	GB
512	318 (62.11%)	3	27	$<$ 1sec	$<$ 1sec	$<$ 1sec
512	302 (58.98%)	6	125	$<$ 1sec	30sec	5sec
512	297 (58.00%)	9	343	2sec	8min	56sec
1024	634 (61.91%)	3	27	$<$ 1sec	1sec	1sec
1024	601 (58.69%)	6	125	$<$ 1sec	39sec	9sec
1024	589 (57.52%)	9	343	3sec	10min	2min
1792	1108 (61.83%)	3	27	$<$ 1sec	1sec	1sec
1792	1051 (58.65%)	6	125	$<$ 1sec	50sec	15sec
1792	1028 (57.37%)	9	343	3sec	13min	3min

n	k (known bits)	m	$\lvert\mathcal{M}\rvert$ (lattice dim.)	Runtime		
				$\mathcal{F}$	LLL	GB
512	438 (85.55%)	2	33	$<$ 1sec	1sec	$<$ 1sec
512	419 (81.84%)	4	165	$<$ 1sec	52sec	4sec
512	405 (79.10%)	6	469	1sec	16min	34sec
1024	874 (85.35%)	2	33	$<$ 1sec	1sec	$<$ 1sec
1024	830 (81.05%)	4	165	$<$ 1sec	1min	6sec
1024	808 (78.91%)	6	469	1sec	22min	57sec
1792	1528 (85.27%)	2	33	$<$ 1sec	2sec	1sec
1792	1451 (80.97%)	4	165	$<$ 1sec	2min	7sec
1792	1412 (78.79%)	6	469	1sec	31min	2min

Table 1. Experimental results for CSIDH / Theorem 6 (top) and CSURF / Theorem 7 (bottom) with n-bit prime p and k-bit MSB oracle, averaged over 10 runs each. The columns m and $\lvert\mathcal{M}\rvert$ show the parameters m and $\lvert\mathcal{M}\rvert$ used in Coppersmith's method. The columns $\mathcal{F}$, LLL and GB show the required runtime for constructing the set $\mathcal{F}$, running LLL and computing the Gröbner basis, respectively. For every n and m, the table shows the smallest k for which our algorithms were able to solve $\mathsf{CI\text{-}HNP}_k$.

and demonstrated its practicality by using it to solve the Commutative Isogeny Hidden Number Problem. In particular, we gave highly simplified proofs for the recovery bound of our algorithm that only rely on a mild heuristic.

Open Problems. Lastly we state some open problems. Improving the recovery bound for either CSIDH or CSURF would of course be desirable. Apart from incremental improvements coming from an improved Coppersmith lattice the only other natural option seems to be to incorporate higher-degree isogenies. This would yield more polynomial relations at the expense of higher total degrees

of said polynomials. It is currently not known whether this trade-off can be used to increase the overall recovery rate. Alternatively, finding a completely different approach to solving CI-HNP would be very intriguing.

Secondly, any improvements to Corollary 4 would be welcome, either by extending the reduction to adversaries with non-negligible success probability or by removing the condition on CSIDH/CSURF being effective group actions.

Thirdly, proving Heuristic 2 (even in some special cases) would be very interesting as it would yield an efficient algorithm that can derive provably correct recovery bounds.

Acknowledgements. We would like to thank Sabrina Kunzweiler for her helpful discussions and pointing us to the 4-isogenies in the CSIDH setting. Jonas Meers was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2092 CASA - 390781972. Julian Nowakowski is funded by the DFG grant 465120249.

References

1. Alamati, N., De Feo, L., Montgomery, H., Patranabis, S.: Cryptographic group actions and applications. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part II. LNCS, vol. 12492, pp. 411–439. Springer, Heidelberg (Dec 2020). https://doi.org/10.1007/978-3-030-64834-3_14
2. Aranha, D.F., Fouque, P.A., Gérard, B., Kammerer, J.G., Tibouchi, M., Zapalowicz, J.C.: GLV/GLS decomposition, power analysis, and attacks on ECDSA signatures with single-bit nonce bias. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014, Part I. LNCS, vol. 8873, pp. 262–281. Springer, Heidelberg (Dec 2014). https://doi.org/10.1007/978-3-662-45611-8_14
3. Boneh, D., Halevi, S., Howgrave-Graham, N.: The modular inversion hidden number problem. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 36–51. Springer, Heidelberg (Dec 2001). https://doi.org/10.1007/3-540-45682-1_3
4. Boneh, D., Shparlinski, I.: On the unpredictability of bits of the elliptic curve Diffie-Hellman scheme. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 201–212. Springer, Heidelberg (Aug 2001). https://doi.org/10.1007/3-540-44647-8_12
5. Boneh, D., Venkatesan, R.: Hardness of computing the most significant bits of secret keys in Diffie-Hellman and related schemes. In: Koblitz, N. (ed.) CRYPTO'96. LNCS, vol. 1109, pp. 129–142. Springer, Heidelberg (Aug 1996). https://doi.org/10.1007/3-540-68697-5_11
6. Breitner, J., Heninger, N.: Biased nonce sense: Lattice attacks against weak ECDSA signatures in cryptocurrencies. In: Goldberg, I., Moore, T. (eds.) FC 2019. LNCS, vol. 11598, pp. 3–20. Springer, Heidelberg (Feb 2019). https://doi.org/10.1007/978-3-030-32101-7_1
7. Castryck, W., Decru, T.: CSIDH on the surface. In: Ding, J., Tillich, J.P. (eds.) Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020. pp. 111–129. Springer, Heidelberg (2020). https://doi.org/10.1007/978-3-030-44223-1_7
8. Castryck, W., Decru, T.: An efficient key recovery attack on SIDH. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part V. LNCS, vol. 14008, pp. 423–447. Springer, Heidelberg (Apr 2023). https://doi.org/10.1007/978-3-031-30589-4_15

9. Castryck, W., Decru, T., Vercauteren, F.: Radical isogenies. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part II. LNCS, vol. 12492, pp. 493–519. Springer, Heidelberg (Dec 2020). https://doi.org/10.1007/978-3-030-64834-3_17
10. Castryck, W., Lange, T., Martindale, C., Panny, L., Renes, J.: CSIDH: An efficient post-quantum commutative group action. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018, Part III. LNCS, vol. 11274, pp. 395–427. Springer, Heidelberg (Dec 2018). https://doi.org/10.1007/978-3-030-03332-3_15
11. Castryck, W., Panny, L., Vercauteren, F.: Rational isogenies from irrational endomorphisms. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part II. LNCS, vol. 12106, pp. 523–548. Springer, Heidelberg (May 2020). https://doi.org/10.1007/978-3-030-45724-2_18
12. Castryk, W.: Csidh on the surface (csurf) (2021), https://homes.esat.kuleuven.be/~wcastryc/summer_school_csurf.pdf
13. Coppersmith, D.: Finding a small root of a bivariate integer equation; factoring with high bits known. In: Maurer, U.M. (ed.) EUROCRYPT'96. LNCS, vol. 1070, pp. 178–189. Springer, Heidelberg (May 1996). https://doi.org/10.1007/3-540-68339-9_16
14. Coppersmith, D.: Finding a small root of a univariate modular equation. In: Maurer, U.M. (ed.) EUROCRYPT'96. LNCS, vol. 1070, pp. 155–165. Springer, Heidelberg (May 1996). https://doi.org/10.1007/3-540-68339-9_14
15. Coppersmith, D.: Small solutions to polynomial equations, and low exponent RSA vulnerabilities. Journal of Cryptology **10**(4), 233–260 (Sep 1997). https://doi.org/10.1007/s001459900030
16. Couveignes, J.M.: Hard homogeneous spaces. Cryptology ePrint Archive, Report 2006/291 (2006), https://eprint.iacr.org/2006/291
17. Dall, F., De Micheli, G., Eisenbarth, T., Genkin, D., Heninger, N., Moghimi, A., Yarom, Y.: CacheQuote: Efficiently recovering long-term secrets of SGX EPID via cache attacks. IACR TCHES **2018**(2), 171–191 (2018). https://doi.org/10.13154/tches.v2018.i2.171-191, https://tches.iacr.org/index.php/TCHES/article/view/879
18. De Feo, L., Jao, D., Plût, J.: Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. Cryptology ePrint Archive, Report 2011/506 (2011), https://eprint.iacr.org/2011/506
19. De Mulder, E., Hutter, M., Marson, M.E., Pearson, P.: Using Bleichenbacher's solution to the hidden number problem to attack nonce leaks in 384-bit ECDSA. In: Bertoni, G., Coron, J.S. (eds.) CHES 2013. LNCS, vol. 8086, pp. 435–452. Springer, Heidelberg (Aug 2013). https://doi.org/10.1007/978-3-642-40349-1_25
20. Duman, J., Hartmann, D., Kiltz, E., Kunzweiler, S., Lehmann, J., Riepel, D.: Group action key encapsulation and non-interactive key exchange in the QROM. In: Agrawal, S., Lin, D. (eds.) ASIACRYPT 2022, Part II. LNCS, vol. 13792, pp. 36–66. Springer, Heidelberg (Dec 2022). https://doi.org/10.1007/978-3-031-22966-4_2
21. Galbraith, S., Panny, L., Smith, B., Vercauteren, F.: Quantum equivalence of the DLP and CDHP for group actions. Cryptology ePrint Archive, Report 2018/1199 (2018), https://eprint.iacr.org/2018/1199
22. Galbraith, S.D., Petit, C., Shani, B., Ti, Y.B.: On the security of supersingular isogeny cryptosystems. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016, Part I. LNCS, vol. 10031, pp. 63–91. Springer, Heidelberg (Dec 2016). https://doi.org/10.1007/978-3-662-53887-6_3

23. Håstad, J.: On using RSA with low exponent in a public key network. In: Williams, H.C. (ed.) CRYPTO'85. LNCS, vol. 218, pp. 403–408. Springer, Heidelberg (Aug 1986). https://doi.org/10.1007/3-540-39799-X_29
24. Herrmann, M., May, A.: Attacking power generators using unravelled linearization: When do we output too much? In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 487–504. Springer, Heidelberg (Dec 2009). https://doi.org/10.1007/978-3-642-10366-7_29
25. Howgrave-Graham, N.: Approximate integer common divisors. In: Silverman, J.H. (ed.) Cryptography and Lattices, International Conference, CaLC 2001, Providence, RI, USA, March 29-30, 2001, Revised Papers. Lecture Notes in Computer Science, vol. 2146, pp. 51–66. Springer (2001). https://doi.org/10.1007/3-540-44670-2_6, https://doi.org/10.1007/3-540-44670-2_6
26. Jao, D., Jetchev, D., Venkatesan, R.: On the bits of elliptic curve Diffie-Hellman keys. In: Srinathan, K., Rangan, C.P., Yung, M. (eds.) INDOCRYPT 2007. LNCS, vol. 4859, pp. 33–47. Springer, Heidelberg (Dec 2007)
27. Jochemsz, E., May, A.: A strategy for finding roots of multivariate polynomials with new applications in attacking RSA variants. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 267–282. Springer, Heidelberg (Dec 2006). https://doi.org/10.1007/11935230_18
28. Kawashima, T., Takashima, K., Aikawa, Y., Takagi, T.: An efficient authenticated key exchange from random self-reducibility on CSIDH. In: Hong, D. (ed.) ICISC 20. LNCS, vol. 12593, pp. 58–84. Springer, Heidelberg (Dec 2020). https://doi.org/10.1007/978-3-030-68890-5_4
29. de Kock, B., Gjøsteen, K., Veroni, M.: Practical isogeny-based key-exchange with optimal tightness. In: Dunkelman, O., Jr., M.J.J., O'Flynn, C. (eds.) SAC 2020. LNCS, vol. 12804, pp. 451–479. Springer, Heidelberg (Oct 2020). https://doi.org/10.1007/978-3-030-81652-0_18
30. Lenstra, A.K., Lenstra, H.W., Lovász, L.: Factoring polynomials with rational coefficients. Mathematische annalen **261**, 515–534 (1982)
31. Maino, L., Martindale, C., Panny, L., Pope, G., Wesolowski, B.: A direct key recovery attack on SIDH. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part V. LNCS, vol. 14008, pp. 448–471. Springer, Heidelberg (Apr 2023). https://doi.org/10.1007/978-3-031-30589-4_16
32. May, A.: Lattice-based integer factorisation: An introduction to coppersmith's method. In: Computational Cryptography: Algorithmic Aspects of Cryptology, p. 78–105. London Mathematical Society Lecture Note Series, Cambridge University Press (2021)
33. May, A., Nowakowski, J., Sarkar, S.: Partial key exposure attack on short secret exponent CRT-RSA. In: Tibouchi, M., Wang, H. (eds.) ASIACRYPT 2021, Part I. LNCS, vol. 13090, pp. 99–129. Springer, Heidelberg (Dec 2021). https://doi.org/10.1007/978-3-030-92062-3_4
34. May, A., Nowakowski, J., Sarkar, S.: Approximate divisor multiples - factoring with only a third of the secret CRT-exponents. In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022, Part III. LNCS, vol. 13277, pp. 147–167. Springer, Heidelberg (May / Jun 2022). https://doi.org/10.1007/978-3-031-07082-2_6
35. Merget, R., Brinkmann, M., Aviram, N., Somorovsky, J., Mittmann, J., Schwenk, J.: Raccoon attack: Finding and exploiting most-significant-bit-oracles in TLS-DH(E). In: Bailey, M., Greenstadt, R. (eds.) USENIX Security 2021. pp. 213–230. USENIX Association (Aug 2021)

36. Montgomery, H., Zhandry, M.: Full quantum equivalence of group action DLog and CDH, and more. In: Agrawal, S., Lin, D. (eds.) ASIACRYPT 2022, Part I. LNCS, vol. 13791, pp. 3–32. Springer, Heidelberg (Dec 2022). https://doi.org/10.1007/978-3-031-22963-3_1
37. Nguyen, P.Q.: The dark side of the hidden number problem: Lattice attacks on dsa. In: Lam, K.Y., Shparlinski, I., Wang, H., Xing, C. (eds.) Cryptography and Computational Number Theory. pp. 321–330. Birkhäuser Basel, Basel (2001)
38. Onuki, H., Takagi, T.: On collisions related to an ideal class of order 3 in CSIDH. In: Aoki, K., Kanaoka, A. (eds.) IWSEC 20. LNCS, vol. 12231, pp. 131–148. Springer, Heidelberg (Sep 2020). https://doi.org/10.1007/978-3-030-58208-1_8
39. Renes, J.: Computing isogenies between Montgomery curves using the action of (0,0). Cryptology ePrint Archive, Report 2017/1198 (2017), https://eprint.iacr.org/2017/1198
40. Robert, D.: Breaking SIDH in polynomial time. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part V. LNCS, vol. 14008, pp. 472–503. Springer, Heidelberg (Apr 2023). https://doi.org/10.1007/978-3-031-30589-4_17
41. Ryan, K., Heninger, N.: Cryptanalyzing MEGA in six queries. Cryptology ePrint Archive, Report 2022/914 (2022), https://eprint.iacr.org/2022/914
42. Ryan, K., Heninger, N.: Fast practical lattice reduction through iterated compression. Cryptology ePrint Archive, Report 2023/237 (2023), https://eprint.iacr.org/2023/237
43. Schoof, R.: Nonsingular plane cubic curves over finite fields. Journal of Combinatorial Theory, Series A **46**(2), 183–211 (1987). https://doi.org/https://doi.org/10.1016/0097-3165(87)90003-3
44. Shani, B.: On the bit security of elliptic curve Diffie-Hellman. In: Fehr, S. (ed.) PKC 2017, Part I. LNCS, vol. 10174, pp. 361–387. Springer, Heidelberg (Mar 2017). https://doi.org/10.1007/978-3-662-54365-8_15
45. Shor, P.W.: Algorithms for quantum computation: Discrete logarithms and factoring. In: 35th FOCS. pp. 124–134. IEEE Computer Society Press (Nov 1994). https://doi.org/10.1109/SFCS.1994.365700
46. Silverman, J.H.: The Arithmetic of Elliptic Curves. Graduate texts in mathematics, Springer, Dordrecht (2009). https://doi.org/10.1007/978-0-387-09494-6, https://cds.cern.ch/record/1338326
47. Takahashi, A., Tibouchi, M., Abe, M.: New Bleichenbacher records: Fault attacks on qDSA signatures. IACR TCHES **2018**(3), 331–371 (2018). https://doi.org/10.13154/tches.v2018.i3.331-371, https://tches.iacr.org/index.php/TCHES/article/view/7278
48. Takayasu, A., Kunihiro, N.: Partial key exposure attacks on CRT-RSA: Better cryptanalysis to full size encryption exponents. In: Malkin, T., Kolesnikov, V., Lewko, A.B., Polychronakis, M. (eds.) ACNS 15. LNCS, vol. 9092, pp. 518–537. Springer, Heidelberg (Jun 2015). https://doi.org/10.1007/978-3-319-28166-7_25
49. Takayasu, A., Lu, Y., Peng, L.: Small CRT-exponent RSA revisited. In: Coron, J.S., Nielsen, J.B. (eds.) EUROCRYPT 2017, Part II. LNCS, vol. 10211, pp. 130–159. Springer, Heidelberg (Apr / May 2017). https://doi.org/10.1007/978-3-319-56614-6_5
50. Takayasu, A., Lu, Y., Peng, L.: Small CRT-exponent RSA revisited. Journal of Cryptology **32**(4), 1337–1382 (Oct 2019). https://doi.org/10.1007/s00145-018-9282-3
51. Vélu, J.: Isogénies entre courbes elliptiques. Comptes-Rendus de l'Académie des Sciences **273**, 238–241 (1971)

52. Wesolowski, B.: Orientations and the supersingular endomorphism ring problem. In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022, Part III. LNCS, vol. 13277, pp. 345–371. Springer, Heidelberg (May / Jun 2022). https://doi.org/10.1007/978-3-031-07082-2_13
53. Xu, J., Hu, L., Sarkar, S.: Cryptanalysis of elliptic curve hidden number problem from pkc 2017. Designs, Codes and Cryptography **88**(2), 341–361 (Feb 2020). https://doi.org/10.1007/s10623-019-00685-y, https://doi.org/10.1007/s10623-019-00685-y
54. Xu, J., Sarkar, S., Wang, H., Hu, L.: Improving bounds on elliptic curve hidden number problem for ECDH key exchange. In: Agrawal, S., Lin, D. (eds.) ASIACRYPT 2022, Part III. LNCS, vol. 13793, pp. 771–799. Springer, Heidelberg (Dec 2022). https://doi.org/10.1007/978-3-031-22969-5_26
55. Yoneyama, K.: Post-quantum variants of ISO/IEC standards: Compact chosen ciphertext secure key encapsulation mechanism from isogeny. In: Proceedings of the 5th ACM Workshop on Security Standardisation Research Workshop. p. 13–21. SSR'19, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3338500.3360336