Analysis of some natural variants of the PKP Algorithm

Rodolphe LAMPE and Jacques PATARIN

PRISM - University of Versailles

In 1989, Adi Shamir [15] proposed a new zero-knowledge identification scheme based on a NP-complete problem called PKP for Permuted Kernel Problem. For a given prime p, a given matrix A and a given vector V, the problem is to find a permutation π such that the permuted vector V_π verifies $A \cdot V_\pi = 0 \mod p$.

This scheme is still in 2011 known as one of the most efficient identification scheme based on a combinatorial problem. However, we will see in this paper that it is possible to improve this scheme significantly by combining new ideas in order to reduce the total number of computations to be performed and to improve very efficiently the security against side channel attacks using precomputations. We will obtain like this a new scheme that we have called SPKP.

Moreover, if we use precomputed values in the scheme SPKP, then the prover will need to perform no computations (i.e. only selection and transmission of precomputed values). This is very interesting for security against side channel attacks because our scheme is zero-knowledge and we don't perform any computations using the key during the identification so we prove that any attacker (even using side channel attacks) being successfully identified implies that he has a solution to the NP-complete problem PKP.

1 Introduction

The articles published on PKP after Adi Shamir's article of 1989 focussed on the study of various attacks on PKP. In 1992, Georgiades [5] introduced symmetric polynomials equations. The symmetric polynomial equation of degree 1 is very useful and will be used by every other attacks. The symmetric polynomial equations of bigger degrees seems to be very difficult to exploit though. The same year, Baritaud, Campana, Chauvaud and Gilbert [1] attacked PKP using a time-memory trade-off. In 1994, Chauvaud and Patarin [2] combined the previous attacks and used a few new ideas. In 1997, Poupard [13] created a program to find the best attack's parameters improving the previous techniques. In 2001, Joux ([8]) used a new time-memory trade-off technique, dividing equations in 4 parts, to further improve the attack.

However, these attacks didn't break Shamir's PKP scheme : they are all exponential and PKP is still very efficient. For example, the best attack known from Joux [8] is in 2^{106}. Nevertheless, they show that the initial parameters of Shamir PKP(16,32) are too weak, specially with today's power computation.

Some articles (Girault [6], Courtois [3]) compared the PKP scheme with other identification scheme like CLE [17] [18] and SD [16] from Stern, PPP [12] from Pointcheval and MQ [14] from Sakumoto/Shirai/Hiwatari. These papers show that PKP scheme is one of the most efficient in terms of computations needed and bits transferred.

In this article, we will try to describe variants of PKP that could make it even more efficient. It seems like this subject has not been studied so far. In fact, as we will see, the simplest variants don't give very good results. In this way, we could say that Shamir's PKP scheme seems quite "stable". Nevertheless we will see that, combining some simple ideas, we can create a scheme, named SPKP, that seems to be really more efficient.

For example, standard parameters PKP(37,64) needs 2^{15} multiplications of 8 bits numbers and 2^{15} additions of 8 bits numbers (for a 2^{106} security and a 2^{-30} impersonation probability) and the number of operations remains the same with a 32 bits microprocessor.
Our new version SPKP needs $2^{14.4}$ additions of 8 bits numbers (still for 2^{106} security against the best known attacks) and $2^{12.4}$ additions of 32 bits numbers if we use a 32 bits microprocessor.

On modern microprocessors, 8 bits additions and 8 bits multiplications cost about the same but it may be interesting to use additions instead of multiplications on very cheap RFID, or when the modulo p becomes large.

We will also see that our scheme SPKP is perfectly safe against SCA (side channel attacks). With PKP, we need $2^{23.4}$ bits of precomputed values to be perfectly safe against SCA, this is not realistic. With SPKP, we need 2^{17} bits of precomputed values to be perfectly safe against SCA, this a major improvement.

Part I - The original PKP (Shamir, 1989)

2 Definitions of PKP and the corresponding identification scheme

Let p be a prime, V a vector of $\mathbb{Z}_p^n$, A a matrix of $\mathbb{Z}_p^{m \times n}$. For each permutation $\sigma \in S_n$, we note V_σ the vector defined by $V_\sigma = (v_{\sigma(i)})_i$ and A_σ the matrix defined by $A_\sigma = (a_{i,\sigma(j)})_{i,j}$. We can notice that, for each permutation σ, we have $A_\sigma R_\sigma = AR$.

Given a prime p, a matrix A and a vector V, the Permuted Kernel Problem is to find a permutation π such that $A.V_\pi = 0 \mod p$.

This problem is NP-complete and has many advantages to be used in an identification scheme. Indeed, the following identification scheme is Zero-Knowledge (the prover doesn't reveal anything about the secret during the identification), it uses very basic operations (multiplications mod p), it is very fast and it differs from many other schemes by not depending of the factorisation or discrete log problem. Since the problem is NP-complete, it is expected to be secure against quantum computers (unlike schemes based on factorisation or discrete log).

The identification scheme is the following :

PKP 5 rounds identification scheme [15]

The users agree on a matrix A and a prime p. Each user chooses a random vector W in $\mathrm{Ker}(A)$, a random permutation π and computes $V = W_{\pi^{-1}}$. The public key will be V and the secret key will be π. V has been defined such that V_π is in $\mathrm{Ker}(A)$. Each user can now prove their identity by proving they know π :

1. The prover chooses a random vector R and a random permutation σ. The prover computes the hashed values of $(\sigma, A.R)$ and $(\pi\sigma, R_\sigma)$ and sends both of them to the verifier.
2. The verifier chooses a random $c \in \mathbb{Z}/p\mathbb{Z}$ and sends it to the prover.
3. The prover sends $W = R_\sigma + cV_{\pi\sigma}$.
4. The verifier sends a bit b.
5. The prover sends σ if $b = 0$ and sends $\pi\sigma$ if $b = 1$.
   In the first case, the verifier checks that the hash of $(\sigma, A_\sigma W)$ is equal to the hash of (σ, AR).
   In the second case, the verifier checks that the hash of $(\pi\sigma, W - cV_{\pi\sigma})$ is equal to the hash of $(\pi\sigma, R_\sigma)$.

An honest prover is obviously passing the test successfully : in the first case, we verify that

$$A_\sigma W = A_\sigma(R_\sigma + cV_{\pi\sigma}) = A_\sigma R_\sigma + cA_\sigma V_{\pi\sigma} = AR + cAV_\pi = AR.$$

In the second case, we verify that

$$W - cV_{\pi\sigma} = R_\sigma.$$

As shown in [15], the scheme is Zero-Knowledge and the probability of success for someone who doesn't know π is less or equal to $\frac{p+1}{2p}$. For 31 iterations, the probability of success is approximately 2^{-30}.

3 Parameters

We'd like to have only one solution for each PKP problem. If there are too many equations, this gives too much information. If there are not enough equations,

there are too many permutations solutions. So, we have to find the good number of equations.

The probability for a random vector to be in the Kernel of A is p^{-m} because there are m equations. The cardinal of the orbit of V under the permutations (ie the set $\{V_\sigma\}_\sigma$) is $n!$ if V has distinct coordinates. In order to have only one solution and to give the good proportion of information, we need to have $n! \approx p^m$. This is the first constraint.

Now, we have to care about security. The naive attack is to choose the first $n-m$ coordinates of the vector V_π (using the coordinates of V) and use the m equations to find the last m coordinates. The complexity of this naive attack is $\frac{n!}{m!}$. We need n to be big enough so that $\frac{n!}{m!}$ is big enough, this is the second constraint.

Later, we will use the best known attack from Joux but in the next sections, we'll only need the naive attack to understand that the simplest variants are not efficient.

Shamir proposed to use $p = 251$ (the largest prime number on 8 bits) so that we can use the scheme on small devices like 8 bits microprocessors of smart cards. This is a good choice and we'll see in section 5 if we can choose other values for p (for example for 32 or 64 bits processors, are larger values of p more efficient ?). Considering the two constraints, values of n and m were proposed : $PKP(16, 32)$ (which gives a security in 2^{46} against the best known attack at present and therefore is not sufficient) and $PKP(37, 64)$ (which gives a security in 2^{106} against the best known attack).

4 Performances

Let's count how many multiplications we need to do in the identification scheme.

The matrix is A is public so everyone can use Gauss elimination so we can assume A is given by $A = [A'|I]$ where A' is a $m \times (n-m)$ matrix and I is the $m \times m$ identity matrix.

The prover has to compute $A.R$ at step 1 and $c.V_{\pi\sigma}$ at step 3. This is $m \times (n-m) + n$ multiplications of 8bits numbers and the same number of additions. For PKP$(16, 32)$, after 31 rounds, this is $2^{14.1}$ operations (half of them are multiplications). For PKP$(37, 64)$, after 31 rounds, this is 2^{16} operations. This is very fast compared to many other schemes.

In each round, we send two hashed values (128 bits for both), one vector ($8n$ bits) and one permutation ($\log_2(n!)$ bits). For $PKP(16, 32)$, after 31 rounds, this is $2^{13.9}$ bits and for $PKP(37, 64)$ this is $2^{14.8}$ bits.

Part II - Analysis of some simple variants of PKP

5 First variant : Different values of p

5.1 Why $2 \leq p < 251$ is not a good choice in PKP

p=2 : There are many issues in using $p = 2$. The first one is that we don't have $n!$ different possible solutions anymore because there are many equal coordinates. The best way to keep many different solutions is to set V with $n/2$ zeros and $n/2$ ones, that way we have $\frac{n!}{(n/2)!^2}$ different possibles solutions.

Moreover, we found another problem with $p = 2$: if two public keys V_1 and V_2 have the same number of ones and zeros, the user knowing π_1 can compute π_2 and inversely. The proof is in the appendice 13.1. This limits the number of possible keys to $n + 1$ at best but most of them are weak.

2<p<251 : For those values of p, we have the same problems that we had with $p = 2$. It's difficult to build public keys with different coordinates and there is a limited number a possible public keys (specially for small values of p). All the details are in the appendice 13.1

5.2 Why $p > 251$ improve the number of operations needed but not the transmissions

Nowadays, we have access to 32 and 64bits processor so we could use those to compute modulo prime numbers of 32 or 64 bits. Therefore it is rather natural to consider PKP on computers (instead of 8 bits smartcards) with values of p of 32 or 64 bits instead of $p = 251$ (8 bits). As we will see, we will improve like this the number of computations (but not the number of transmissions).

The equation $n! \approx p^m$ tell us that, using 4 or 8 times more bits for p, divides m by 4 or 8. We know that all the attack heavily use those m equations (for example, the naive attack is in $n!/m!$).

Let see in the next array some parameters, the corresponding Joux's attack complexity and the number of multiplications mod p needed. We can notice that the extra equation of degree one of Georgiades [5] is used here in Joux attack and is not negligible when p is large. We made different arrays for the various ranges of attack's complexity (cf appendice 13.2 for more details).

p	m	n	Security	Operations	Transmissions
251	24	46	2^{79}	$2^{15.1}$	$2^{14.4}$
2^{16}	8	34	2^{80}	$2^{13.8}$	$2^{14.6}$
2^{32}	4	34	2^{104}	$2^{13.2}$	$2^{15.3}$
2^{64}	2	34	2^{80}	$2^{12.5}$	$2^{16.2}$

Security in 2^{80}

p	m	n	Security	Operations	Transmissions
251	34	60	2^{102}	$2^{15.8}$	$2^{14.7}$
2^{16}	10	40	2^{106}	$2^{14.4}$	$2^{14.8}$
2^{32}	4	34	2^{104}	$2^{13.2}$	$2^{15.3}$
2^{64}	3	46	2^{151}	$2^{13.4}$	$2^{16.6}$

Security in 2^{100}

For p with $8, 16, 32$ or 64 bits, we gave the parameters that gives a security greater or equal to 2^{80} for the first array and 2^{100} for the second one. We remark that, for big values of p, the choice of m is very limited so we can't adjust very well the parameters to reach a security close to 2^{80} or 2^{100}.

Those results tell us that using big values of p permits to reduce significantly the number of operations. As shown above, with a 2^{80} security, it needs $2^{12.5}$ operations for p with 64 bits, $m = 2, n = 34$ and needs $2^{15.1}$ operations for the standard PKP(24,46). This is 6 times faster.

However, the number of transmissions are bigger with big values of p. Comparing the same parameters than above, we go from $2^{14.4}$ bits to $2^{16.2}$, this is 3.5 times bigger.

6 Second variant : 3 rounds PKP (instead of 5 round PKP)

We found an identification scheme with 3 rounds (instead of 5) based on the PKP problem :

PKP 3 rounds identification scheme

1. The prover chooses a random vector R and a random permutation σ. He sends 4 hashed values :

$$h_1 = H(\sigma), \quad h_2 = H(R_\sigma + (V_\pi)_\sigma), \quad h_3 = H(AR), \quad h_4 = H(R_\sigma)$$

2. The verifier sends a challenge $b = 1, 2$ or 3.

3. – If $b = 1$, the prover reveals σ and $W_1 = R_\sigma$.
   The verifier verifies that $H(\sigma) = h_1, H(A_\sigma(W_1)) = h_3$ and $H(W_1) = h_4$.

   – If $b = 2$, the prover reveals σ and $W_2 = R + V_\pi$.
   The verifier verifies that $H(\sigma) = h_1, H((W_2)_\sigma) = h_2$ and $H(AW_2) = h_3$.

   – If $b = 3$, the prover reveals $\sigma\pi$ and $W_3 = R_\sigma$.
   The verifier verifies that $H(W_3 + V_{\sigma\pi}) = h_2$ and $H(W_3) = h_4$.

Theorem :
An honest prover will pass the test successfully all the time while a dishonest prover has, at best, a probability $\frac{2}{3}$ to pass the test successfully. Indeed, if someone can answer to the 3 questions then he has a solution for the PKP problem.

Proof : Let $\sigma_1, W_1, \sigma_2, W_2, \sigma_3$ and W_3 be the answers to the 3 possible questions. Since the attacker is accepted for each questions, we have the following system :

$$\begin{cases} H(\sigma_1) & = h_1 \\ H(A_{\sigma_1}(W_1)) & = h_3 \\ H(W_1) & = h_4 \\ H(\sigma_2) & = h_1 \\ H((W_2)_{\sigma_2}) & = h_2 \\ H(AW_2) & = h_3 \\ H(W_3 + V_{\sigma_3}) & = h_2 \\ H(W_3) & = h_4 \end{cases}$$

Supposing that the H functions are secure, we have :

$$\begin{cases} \sigma_1 & = & \sigma_2 \\ A_{\sigma_1}(W_1) & = & AW_2 \\ W_1 & = & W_3 \\ (W_2)_{\sigma_1} & = & W_1 + V_{\sigma_3} \end{cases}$$

Replacing W_1 in the second equation by $(W_2)_{\sigma_1} + V_{\sigma_3}$ (using the fourth equation), we have :

$$A_{\sigma_1}((W_2)_{\sigma_1} - V_{\sigma_3}) = AW_2.$$

Since $A_{\sigma_1}((W_2)_{\sigma_1}) = AW_2$ and $V_{\sigma_3} = (V_{\sigma_1^{-1}\sigma_3})_{\sigma_1}$, this implies that :

$$A_{\sigma_1}(V_{\sigma_1^{-1}\sigma_3})_{\sigma_1} = 0,$$

which is equivalent to $AV_{\sigma_1^{-1}\sigma_3} = 0$. So $\sigma_1^{-1}\sigma_3$ is solution of PKP and the attacker knows the secret. □

Considering this probability, we need 52 rounds to have a 2^{-30} impersonation probability.

Theorem :
The PKP 3-rounds scheme is Zero-Knowledge.

Proof : There are 3 differents answers. If $b = 1$, we reveal σ and R_σ so we don't reveal anything about the secret π. If $b = 2$, we reveal σ and $R + V_\pi$, the secret V_π is not revealed because R is random so $R + V_\pi$ is random. If $b = 3$, we reveal $\sigma\pi$ and R_σ but, again, σ and R being random we don't reveal anything about π. □

About performances, we need to compute AR and $R_\sigma + (V_\pi)_\sigma$ at step 1. This is $m(n-m)$ multiplications and $m(n-m)+n$ additions. About transmissions, there are 4 hash (256 bits), one vector ($8n$ bits) and one permutation ($\log_2(n!)$ bits).

7 Third variant : using more vectors and some symmetry in the PKP problem

What happens if we use more than one vector V i.e. with $l \geq 2$ vectors V instead of $l = 1$? This is the exact same thing to consider V as a matrix of size $m \times l$. If we increase the number of vectors, we have to decrease the number of equations m (because $n! \approx p^{m \cdot l}$) which should increase the attack complexity. Here we will explain why $l = m$ (the maximal possible value for l) is not more secure than $l = 1$ (they actually have the exact same security, using a symmetry argument). However, in section 8 we will see that $l = \sqrt{m}$ is indeed interesting because, in this situation, there is an equal number of equations and vectors which is the fixed point of the following symmetry :

Theorem (Symmetry in PKP) :
Given a prime p and an integer n, solving the PKP problem with m equations and l vectors has the exact same complexity that solving the PKP problem with l equations and m vectors.

Proof in Appendice 13.5.

As said before, decreasing m increase the attack's complexity but it slightly increase the number of operations needed as well. Using p of $8, 16, 32$ or 64 bits didn't lead to interesting results. But, surprisingly, using $p = 2$ and multiple vectors permits to decrease the number of operations as we will see in the next part.

Part III - SPKP : our new PKP variant

Our new idea :
With one vector, as we have seen above, we couldn't use low prime numbers. However, with multiple vectors, we noticed that it is now possible to use low prime numbers and we will see that it can be interesting to consider $p = 2$. As long V doesn't have two equal lines, there are $n!$ possibilities for π.

8 Definition

Our new scheme SPKP combines the three previous ideas :

- 3 rounds (instead of 5 for PKP).
- $p = 2$ (instead of $p = 251$ for Shamir's PKP recommended parameters).
- multiple vectors (typically $l = 9$ instead of $l = 1$ for PKP).

Considering multiple vectors, we studied what's the best choice for p and it seems like $p = 2$ gives the best results. Using the 3 rounds scheme gives better results as well. At step 1, to compute h_2, we have to do $n \times l$ additions. At step 1, to compute h_3, we have to compute $A.R$ which has $m \times l$ coordinates. For each of them, we have to do $\frac{n-m}{2}$ bits additions. Indeed, the last m coordinates of each line of A has $m-1$ zeros and 1 one, half of the other coordinates are zeros.

At step 3, when $b = 2$, we need $\frac{nl}{2}$ bits additions to compute W_2.

The total is $\frac{m \times l \times (n-m)}{2} + \frac{3 \times n \times l}{2}$ bits additions.

We need 52 rounds to have a 2^{-30} impersonation probability.

We need to compute the best parameters n, m and l for SPKP and we'll see the number of additions needed and conclude on its potential efficiency. To find those parameters, we have to analyze how SPKP is resisting to the best attack known. This is the object of the next section.

9 Attacks and Efficiency of SPKP

SPKP is NP-complete because PKP is NP-complete and PKP is a particular case of SPKP. SPKP is Zero-Knowledge, the proof is exactly the same we gave for PKP 3 rounds. We think that all existing attacks are less efficient on SPKP. Indeed, for given values n and p, we have the equation

$$n! = p^{m \times l},$$

where m is the number of equations and l is the number of vectors. If we use l vectors instead of 1, we have to divide the number of equations m by l and every attacks are very much dependent of the number of equations m.

The best attack on SPKP seems to be, as far as we know, similar to the best attack on PKP such as Joux's attack [8]. It'd be too long and complicate to describe the Joux's attack there. We adjusted it to SPKP by taking $p = 2$, changing $n-$vectors in $n \times l-$matrix and numbers (in the D_i sets) in $1 \times l-$vectors. In the next arrays, we present the PKP parameters and the SPKP parameters with the corresponding attack's complexity and the number of computations needed for a 2^{-30} impersonation (31 rounds for PKP and 52 rounds for SPKP). Nowadays, recent smart cards use 32 bits microprocessor so we combined bits operations together to divide the number of operations needed.

Parameters	Security	32 bits operations
$SPKP(15, 38, 10)$	2^{79}	$2^{11.9}$ additions
$PKP(24, 46)$	2^{79}	$2^{15.1}$ operations

2^{80} complexity

Parameters	Security	32 bits operations
$SPKP(15, 42, 11)$	2^{98}	$2^{12.2}$ additions
$PKP(34, 60)$	2^{102}	$2^{15.8}$ operations

2^{100} complexity

This results shows that SPKP needs less and simpler operations so it seems to be more efficient than the original PKP scheme. For a 2^{100} security, SPKP needs 12 times less operations than PKP and all operations are additions compared to PKP using multiplications and additions.

Now, we will compare SPKP with other combinatorial schemes. In this array, we'll show bits operations for SPKP instead of combining them and we give the number field used as well. We used parameters for a 2^{-80} security and 2^{-30} impersonation probability.

Schemes	CLE	SD	PP	MQ	PKP	SPKP
Operations	$2^{15}/\mathbb{F}_{257}$	$2^{18}/\mathbb{F}_{256}$	$2^{21}/\mathbb{F}_{127}$	$2^{26}/\mathbb{F}_2$	$2^{15}/\mathbb{F}_{251}$	$2^{17}/\mathbb{F}_2$

This shows that SPKP is the scheme using the less operations (if we combine bits operations together).

Moreover, using precomputations, we can make this scheme even more efficient as we will see in the next section.

10 Precomputations with SPKP or our PKP 3-rounds

In the original PKP scheme (5 rounds, presented at section 2), the prover has to compute $W = R_\sigma + cV_{\pi\sigma}$ where c is a value with p possibilities ($p = 251$ typically) chosen by the verifier then he will face one of the two challenges. Therefore, for the 31 rounds, if the prover want to precompute all the possible answers to the prover questions in advance, he has to prepare 62p answers, approximately 15000 values. This is not very realistic.

However, in our scheme (PKP 3 rounds, section 6), the verifier will face one of the three challenges at each round. Therefore, for the 52 rounds, the prover has to prepare 104 answers for one identification. This is realistic if we use devices with enough memory.

Therefore, we see that all the prover's computations can be precomputed so that the prover doesn't have to compute anything during the identification. We can create a smart card which contains only the datas $R_i, h_1, ..., W_1, ...$ The prover uses this card for identification, his only need is to send and receive datas from the verifier.

Precomputation with other combinatorial schemes

This property is possible on every other schemes as long the number of possible challenges is limited. This is why our scheme is efficient for precomputation : there are only 3 possible challenges while the standard PKP has 502 possible challenges. The memory needed for one identification is the number of bits for transmission times the number of challenges.

In the following array, we show how much memory is needed for one identification using precomputations (with 2^{80} security and 2^{-30} impersonation probability).

Schemes	MQ	SPKP	SD	PP	PKP	CLE
Memory needed in bits	$2^{16.4}$	$2^{16.9}$	$2^{17.5}$	$2^{18.6}$	$2^{23.4}$	$2^{23.8}$

We see that SPKP is one of the most efficient scheme if we want to use precomputation. In 2004, Samsung realised a smart card with 256kbytes of EEPROM which permits to save datas for about 16 identifications using SPKP. In the next few years, the memory size augmentation could permit to create smart cards with more than a thousand of identifications saved.

11 Security against side channel attacks

Since a few years, very efficient physical attacks have been discovered on smart cards and microprocessors, for example : timing attacks, power attacks (SPA, DPA [9]), fault attacks (DFA), ... Generally some ways to fix those problems was found by the scientific community, but sometimes it is really difficult to design secure hardwares against some physical attacks, and it is expected that new attacks could be found. A lot of those attacks use the fact that the microprocessor has to manipulate secret datas. In the variants of PKP 3 rounds and SPKP that we presented, it is possible to precompute everything. That way, no secret datas are manipulated by the microprocessor which greatly simplifies the security against physical attacks.

The precomputed datas have to be encrypted or saved in protected areas because, even if a single data doesn't reveal anything about the secret, the combination of some datas could reveal the secret. The microprocessor needs to be able to transmit one of those values but not all of them and eventually decrypt this value with a key K. The other values have to be encrypted with different keys or saved in protected areas to assure a good security. In fact, it seems to be much easier to secure such a scheme from physical attacks than to secure the traditional schemes that manipulate a secret data s in the computation of an identification against physical attacks (where s needs to be still secret after the identification). That's why we think those schemes present a real interest for the security against physical attacks.

We compared SPKP with other schemes that use precomputations like GPS [7] or Lamport [10] and his variants. We give more details in appendice 13.3.

12 Conclusion

In this article, we studied simple variants of PKP. Using one idea alone doesn't give good results but, surprisingly, combining 3 ideas creates a more efficient

scheme. Those 3 ideas are : 1. to change the characteristic to 2, 2. to use multiple vectors and 3. to use a 3 rounds scheme instead of 5 rounds. As far as we know, it is the first time that a simple 3 round variant of PKP (instead of 5 rounds of [15]) is described.

This new scheme called SPKP has good performance. On a 32 bits processor and for a 2^{80} security, SPKP needs $2^{11.9}$ additions compared to $2^{15.1}$ operations (half of them being multiplications) for PKP. On a 32 bits processor and for a 2^{100} security, SPKP needs $2^{12.2}$ additions compared to $2^{15.8}$ operations (half of them being multiplications) for PKP.

Moreover, all the computations can be precomputed so that the prover doesn't have to compute anything during the identification i.e. he needs only to select some precomputed values and send them. Since no secret value is used or loaded on the micro-processor during this identification this property might be very useful for security against the side channel attacks. This may be really interesting since side channel attacks are often much more efficient than algorithmic attacks for practical security. Typically if two stored and precomputed values are given the secret may be found but only one of this value will be given, in a zero-knowledge way. We can notice that this property is much more efficient on our 3 round variant of PKP and SPKP than the previous classical 5 round variant.

Acknowledgments
The PhD of Rodolphe LAMPE is financially supported by the Direction Générale des Armées (DGA).

References

1. **T. Baritaud, M. Campana, P. Chauvaud and H. Gilbert**. On the security of the Permuted Kernel Identification Scheme. *Crypto '92*. Springer-Verlag.
2. **P. Chauvaud and J. Patarin**. Improved Algorithms for the Permuted Kernel Problem. In D. R. Stinson, editor, *Advances in Cryptology - Proceedings of CRYPTO '93*, volume 773 of *Lecture Notes in Computer Science*, pages 391-402. Springer-Verlag, 1994.
3. **N. Courtois**. Efficent zero-knowledge authentication based on a linear algebra problem MinRank. *ASIACRYPT 2001. Lecture Notes in Computer Science 2248.* Springer 2001, pp. 402-421
4. **A. Fiat and A. Shamir**. How to prove yourself : Practical solutions to identification and signatures problems. *CRYPTO '86. Lectures Notes in Computer Science 263*. Springer 1987, p186-194.
5. **J. Georgiades**. Some Remarks on the security of the Identification Scheme Based on Permuted Kernels. *Journal of Cryptology*, 5:133-137, 1992.
6. **M. Girault**. A survey of identification schemes. *EUROCODE '90. Lecture Notes in Comput. Sci.*, p167-179. Springer, 1991.
7. **M. Girault, G. Poupard, J. Stern**. On the fly authentication and signature schemes based on groups of unknown order. *Journal of Cryptology nÂ° 19*. pp. 463-487, 2006.

8. **E. Jaulmes and A. Joux**. Cryptanalysis of PKP: A new approach. In K. Kim, editor, *Public Key Cryptography*, volume 1992 of *Lecture Notes in Comput. Sci.*, pages 165-172. Springer, 2001.
9. **P. Kocher, J. Jaffe, B. Jun**. Differential Power Analysis. *CRYPTO '99. Lectures Notes in Computer Science 1666.* pp. 388-397.
10. **L. Lamport**. Constructing digital signatures from a one-way function, *Communications of the ACM 1981.* Volume 24, issue 11.
11. **R. Merkle**. A digital signature based on a conventional encryption function, *CRYPTO '87, Lecture Notes in Computer Science 293.* pp. 369-378.
12. **D. Pointcheval**. A new identification scheme based on the perceptrons problem. *EUROCRYPT '95. Lecture Notes in Computer Science 921.* Springer 195, pp. 319-328.
13. **G. Poupard**. A realistic security analysis of identification scheme based on combinatorial problems. *European transactions on telecommunications*, 8:471-480, 1997.
14. **Koichi Sakumoto, Taizo Shirai and Harunaga Hiwatari**. Public-Key Identification Schemes Based on Multivariate Quadratic Polynomials. *Advances in Cryptology*, CRYPTO 2011, 31st Annual Cryptology Conference, volume 6841, p703.
15. **A. Shamir**. An Efficient Identification Scheme Based on Permuted Kernels. In G. Brassard, editor, *Advances in Cryptology - Proceeding of CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, page 606-609. Springer-Verlag, 1990.
16. **J. Stern**. A new identification scheme based on syndrome decoding. *CRYPTO 93. Lecture Notes in Computer Science 773.* Springer 1994, p13-21.
17. **J. Stern**. An alternative to the Fiat-Shamir protocol. *EUROCRYPT '89. Lecture Notes in Computer Science 434.* Springer 1990, pp. 173-180.
18. **J. Stern**. Designing identification schemes with keys of short size. *CRYPTO '94. Lectures Notes in Computer Science 839.* Springer 1994, pp. 164-173.

13 Appendices

13.1 Why small values of p are inefficient

There are many issues in using $p = 2$. The first one is that we don't have $n!$ different possible solutions anymore because there are many equal coordinates. The best way to keep many different solutions is to set V with $n/2$ zeros and $n/2$ ones, that way we have $\frac{n!}{(n/2)!^2}$ different possibles solutions.

Moreover, we found another problem with $p = 2$: if two public keys V_1 and V_2 have the same number of ones and zeros, the user knowing π_1 can compute π_2 and inversely. The proof is in the appendice. This limits the number of possible keys to $n + 1$ at best but most of them are weak.

Indeed, this user just need to compute the permutation σ which sends V_2 to $(V_1)_{\pi_1} \in \mathrm{Ker}(A)$ (this permutation exist because V_2 and V_1 have the same coordinates). That way, we have $(V_2)_\sigma = (V_1)_{\pi_1}$ so

$$A(V_2)_\sigma = A(V_1)_{\pi_1} = 0.$$

This implies that $\pi_2 = \sigma$.

We have, at best, $n+1$ different possible public keys but most of them are not good to use (if there are "too many" zeros or "too many" ones, there are less possible permutations) so using $p = 2$ is inefficient.

p>2 :
We've seen previously that it's much better to have different coordinates for V so that we can have some control on the number of different solutions for π. To construct a vector V with different coordinates, we can just randomly choose the first $n-m$ coordinates using different coordinates then compute the last m coordinates using the m equations. These m last coordinates have to take different values. there are $p-(n-m)$ choices for the first of the last m coordinates, $p-(n-m+1)$ for the next one and so one until the last one having $p-(n-1)$ choices. This gives a probability of :

$$\frac{(p-n+m)!}{(p-n)!p^m}.$$

We want this probability to be big enough so that it doesn't take too much time to build a key. Using the equality $m = log_p(n!)$, we have to study the formula :

$$\frac{(p-n+log_p(n!))!}{(p-n)!p^{log_p(n!)}} = \frac{(p-n+log_p(n!))!}{(p-n)!n!}$$

In the same time, we need $\frac{n!}{m!}$ to be big enough (at least 2^{80}).

With those contraints, there is no solution for p with 5 bits or less. With $p = 61$, the probability of building a key is 2^{-66} with $n = 58$. This is not acceptable. With $p = 127$, the probability goes down to $2^{-6.9}$ with $n = 37$. We could use PKP with $p = 127$ and build "good" keys but it doesn't improve the standard parameters $p = 251$. It's slightly worse because m is bigger with smaller values of p.

We can also use the same argument used for $p = 2$ to attack the other small values of p. We can create only $\frac{(n+p-1)!}{n!(p-1)!}$ public keys. Therefore it is not a good idea to use small values of p.

13.2 Maple computations

We adapted the best known attack on PKP from Joux [8] in order to attack $p = 2$ with multiple vectors.

```
attaque := proc (m, n, l, p)
local n1, n2, n3, n4, k, psi, timeattaque;
n3 := floor((1/4)*n-(1/4)*m-1/4);
n1 := floor((1/3)*n-(1/3)*m-1/3-(1/3)*n3);
n2 := floor((1/2)*n-(1/2)*m-1/2-(1/2)*n1-(1/2)*n3);
n4 := n-m-1-n1-n2-n3;
```

```
k := round(log(factorial(n-n1)/factorial(n))/log((n-n1-n2)/p^l));
psi := ((n-n1-n2)/p^l)^k;
timeattaque := round(log[2]((1/64)*psi*
factorial(n)^2/(factorial(n-n1-n2)*factorial(n-n3-n4))));
return timeattaque
end proc
```

13.3 Comparison with Lamport and GPS

There are very efficient schemes on smart cards. We already cited the combinatorial schemes like CLE [18], SD [16], ... The scheme GPS (Girault Poupard Stern [7]) is also very efficient. It only needs to do one addition during the identification if we have some precomputed values. Nevertheless, this addition manipulates a secret value to protect during the computation and the security of GPS is based on the discrete log problem (attackable on hypothetical quantum computers) which is not a NP-complete problem.

The Lamport's scheme [10] and his variants [11] are also efficient public-key identification schemes. However, if we don't use precomputations, SPKP can realise an arbitrary number of identifications while Lamport is limited. If we use precomputations and we want to do no computations at all during the identification, Lamport seems to be less efficient than SPKP and the public-key size is in $O(n)$ (it was $O(ln(n))$ if we accept computations during the identification), n being the number of identifications needed. A deeper study is in progress.

13.4 Small prime numbers with multiple vectors

We'll show here a toy example explaining why we can use small prime numbers with multiple vectors. In the following example, we see that, with one vector, there are some permutations that doesn't change the permuted vector. If we permute the two first line, the first vector is not modified while the second matrix is modified.

$$\underset{\frac{4!}{2!^2} \text{ permutations}}{\begin{pmatrix}0\\0\\1\\1\end{pmatrix}} \qquad \underset{4! \text{ permutations}}{\begin{pmatrix}0&0\\0&1\\1&0\\1&1\end{pmatrix}}$$

In order to be able to build public-keys with small prime numbers using multiple vectors, we need that there are no two equal lines in V. This is possible as long $p^l \geq n$. With SPKP, we have $p^l \approx 2^{11}$ and $n \leq 2^6$ so, there is no problem to build public keys.

13.5 Symmetry in PKP

If we consider V with l columns instead of 1, we will increase the attack's complexity by reducing m. Indeed, a random vector is in the Kernel of A with a probability $p^{-m \times l}$ so, to keep the same number of solutions, multiplying by l the number of vectors implies to divide by l the number of equations m. We know that all the attacks are more efficient when m increases (for example, the naive attack is $\frac{n!}{m!}$). We could be tempted to multiply the number of vectors by m so we could have only one equation. This is actually useless, let's see why.

Theorem (Symmetry in PKP) :
Given a prime p and an integer n, solving the PKP problem with m equations and l vectors has the exact same complexity that solving the PKP problem with l equations and m vectors.

There is some symmetry for the PKP Problem. We can see the vector V_π as the result of $M_\pi.V$ where M_π is the matrix associated to the permutation π : $M = (\delta_{\pi(i),j})$. That way, the PKP Problem can be written as :

$$A.M_\pi.V = 0.$$

Taking the transpose, we have :

$$^tV.\ ^tM_\pi.\ ^tA = 0.$$

The transpose of M_π is $M_{\pi^{-1}}$ so :

$$(\ ^tV).(\ ^tA)_{\pi^{-1}} = 0.$$

To summarize :

$$A.V_\pi = 0 \Leftrightarrow (\ ^tV).(\ ^tA)_{\pi^{-1}} = 0.$$

Using this symmetry, we know that switching the number of equations with the number of vectors doesn't change the problem.