On the Complexity of the Permuted Kernel Problem

Eliane KOUSSA[1], Gilles MACARIO-RAT[2], and Jacques PATARIN[3]

[1] Versailles Laboratory of Mathematics, UVSQ EJKoussa@outlook.com
[2] Orange gilles.macariorat@orange.com
[3] Versailles Laboratory of Mathematics, UVSQ, CNRS, University of Paris-Saclay, jpatarin@club-internet.fr

Abstract. In this document, we investigate the complexity of an old-time combinatorial problem - namely Permuted Kernel Problem (PKP) - about which no new breakthrough were reported for a while. PKP is an NP-complete [2] algebraic problem that consists of finding a kernel vector with particular entries for a publicly known matrix. It's simple, and needs only basic linear algebra. Hence, this problem was used to develop the first Identification Scheme (IDS) which has an efficient implementation on low-cost smart cards [1].
The Permuted Kernel Problem has been extensively studied [7,8,9,5,10]. We present a summary of previously known algorithms devoted to solve this problem and give an update to their complexity. Contrary to what is shown in [10], and after a thorough analysis of the State-of-the-art attacks of PKP, we claim that the most efficient algorithm for solving PKP is still the one introduced by J. PATARIN and P. CHAUVAUD in [9]. We have been able to specify a theoretical bound on the complexity of PKP which allows us to identify hard instances of this latter.
Among all the attacks, the problem PKP is in spite of the research effort, still exponential.

Keywords: Cryptanalysis · Identification scheme · complexity · post-quantum cryptography · Permuted Kernel Problem.

1 Introduction

In this paper we focus on the analysis of the Permuted Kernel Problem (PKP). In 1989, A. SHAMIR [1] introduced a scheme of a new nature, a Zero-Knowledge (ZK) Identification scheme, based on the Permuted Kernel Problem. PKP is the problem of finding a permutation of a known vector such that the resulting vector is in the kernel of a given matrix. It was proved to be an NP-complete combinatorial problem [1].
PKP requires simple operations which involve basic linear algebra computations. Due to its simplicity, the problem has received significant attentions from theory and application in cryptography. Here, we study the theoretical analysis behind the PKP problem over a finite field. For a little long time, no new attacks on PKP were reported which makes the construction of schemes based on hard instances of this problem more applicable.
We are essentially concerned about this problem because it can be used to build a post-quantum signature scheme based on the hardness of solving random instances of PKP.

In fact and due to the call for post-quantum standards of the NIST, there has been renewed interest in the transformed Zero-Knowledge Identification Schemes into Digital Signatures Schemes (DSS) via the Fiat-Shamir paradigm [3]. This transformation method is important since it yields to efficient signature schemes in terms of minimal and sufficient security assumptions.
Therefore, it is important to reconsider some NP-Complete problems (for example PKP) whose security relies on the fact that there is no quantum algorithms known to solve such problems [4].

Previous works and Main results Since quantum computers are known to be incapable to solve *NP*-Complete problems [4], algebraic problems such PKP, are very interesting nowadays.
The main contribution of this paper is to present an updated analysis of the complexity of the most efficient algorithm for solving instances of the Permuted Kernel Problem.
In [7], J. GEORGIADES proposed the first algorithm to solve PKP. The author presents symmetric polynomials equations which will be utilized by all the other attacks.
The authors of [8] investigate also the security of PKP, where a time-memory trade-off was introduced. Moreover, J. PATARIN and P. CHAUVAUD improve algorithms for the Permuted Kernel Problem[9]. Also, in [10], a new time-memory trade-off was proposed and believed to be the most efficient attack against PKP.
All the suggested attacks combine exhaustive search with some form of time-memory tradeoff.
After our complexity analysis of the Permuted Kernel Problem, it appears that the algorithm of PATARIN-CHAUVAUD [9] is the most efficient one against PKP among them all. Using a Magma code, we have found that the complexity of the algorithm introduced by A. JOUX and E.JAULMES is wrongly estimated. Thus, in order to construct a new public-key cryptosystem based on PKP, we have to define its security in terms of the PATARIN-CHAUVAUD 's algorithm.

2 The Permuted Kernel Problem

In this section, we first present the PKP problem [1] over a finite field $\mathbb{F}_p$.

2.1 Description of PKP

PKP [1,2] is the problem on which the security of the Identification scheme proposed by A. SHAMIR is based. PKP is a linear algebra problem which asks to find a kernel vector of given matrix under a vector-entries constraint. It's a generalization of the Partition problem [2, pg.224]. More precisely, it is defined as follows:

Input. A finite field $\mathbb{F}_p$, a matrix $A \in \mathcal{M}_{m\times n}(\mathbb{F}_p)$ and a n-vector $V \in {\mathbb{F}_p}^n$.
Question. Find a permutation π over $(1,\ldots,n)$ *such that* $A \times V_\pi = 0$, *where* $V_\pi = (V_{\pi(j)}), j = 1,\ldots,n$.

2.2 Practical and complexity considerations

First, for the problem to be hard, $n-m$ must be large enough so that the kernel of A has sufficiently enough elements. Then, since the problem is unchanged by manipulation

on lines of A, one may assume that the matrix A is of rank exactly m. Otherwise an equivalent matrix of A could be expressed with fewer significant lines.

By denoting $A_\sigma = (a_{i\sigma(j)})$, the effect of a permutation σ over the columns of A, it's easy to see that $A_\sigma V_\sigma = AV$. Also, up to a reordering of the columns of A, we may assume that it is given in a systematic form:

$$A = (a_{ij})_{1\le i\le m, 1\le j\le n} \;=\; \big[A'|I\big],$$

where $A' = (a'_{ij})_{1\le i\le m, 1\le j\le n-m} \in \mathcal{M}_{m\times n-m}(\mathbb{F}_p)$ and I is the identity matrix of size m.

Note that, to reach higher security levels, it's more desirable that the n-vector V has distinct coordinates, since it reduce the number of solutions, or otherwise said, it enlarges the search space. If the vector V is drawn at random in the kernel of A and is expected to have this property, then one should draw at random the first distinct $n-m$ variables, computes the m last ones and checks if suitable. So one should have $\frac{(p-n+m)!}{p^m(p-n)!}$ high.

Also, it is most suitable that $p^m \approx n!$ so that the number of solutions is close to 1.

A reduction of the 3-Partition problem proves PKP to be NP-Complete [2] in the good reasoning (*i.e.*its hardness grows exponentially with p). The solidity of PKP comes from, on one hand, the big number of permutations, on the other hand, from the small number of possible permutations which may suit the kernel equations. More precisely, PKP is hard because it obligates the choice of a vector, with already fixed set of entries, from the kernel of the matrix A.

3 Solving PKP: best known algorithms

The implementation's efficiency of the first IDS, proposed by A. SHAMIR [1], based on PKP problem has led to several solving tools, which are all exponential.

As a reference, we mention the exhaustive search consisting in examining all the possible permutations. Its complexity is obviously $n!$.

3.1 J. GEORGIADES method

J. GEORGIADES [7] was the first to publish an improvement of the exhaustive search. The basic idea is to find some new equations in order to reduce the set of suitable permutations. Since the rank of A is equal to m, there exists m linear equations that enable to express the last m coordinates of V_π in terms of the $n-m$ first ones. Therefore, this will decrease the number of permutations to be considered from $n!$ to:

$$\frac{n!}{\big(n-(n-m)\big)!} = \frac{n!}{m!}.$$

Moreover, since the coordinates of V_π are known, all their symmetrical expressions are also known, for example their sum, the sum of their squares, etc. This immediately gives new equations satisfied by the coordinates, whatever the permutation π. Taking into account equations of degree 1 (the sum) and 2 (sum of squares) for instance, this

enables to express $m+2$ variables in terms of the $n-m-2$ others. Therefore again the complexity can be decreased to $n!/(m+2)!$. So far, it is not obvious how to use symmetric equations with higher degree, and decrease more the complexity.

3.2 A time-memory trade-off

An almost simultaneous improvement was introduced by T. BARITAUD, M. CAMPANA, P. CHAUVAUD and H. GILBERT in [8]. The proposed method reduces the time required to solve the PKP problem at the cost of use of significantly large memory. The idea for speeding up the solving time is to perform a pre-computation on a smaller search space involving an A-sub-matrix and the corresponding subsystem. Then naturally in [9], J. PATARIN and P. CHAUVAUD combine this idea with J. GEORGIADES's one, adding the "free" linear symmetric equation, resulting in a reduction of time. They present also some new ideas in order to reduce the needed memory. We may cite:

- "Set introduction" : make an exhaustive search on a sub-set of values, instead of ordered tuples of values. This leads to diminish the size of the initial pre-computation.
- "Middle values" : make an exhaustive search on the value of a sub-system,
- "Pre-computation on A" : search if a sub-system could be expressed with less variables. This leads to probabilistic algorithms.

G. Poupard, in [5] gives a nice generalization of the "Middle values" method, an a corresponding complexity analysis, but that may be flawed since the details of the complexity are not clearly given.

3.3 JOUX-JAULMES algorithm

In [10], A. JOUX and E. JAULMES introduce a new time-memory trade-off algorithm which is an application of the algorithm described in [11] to the Permuted Kernel Problem. This technique includes the so-called 4SET problem (see [12,10] for more details).

Note that, in this section we use the notations of [10].

In fact, the algorithm of JOUX-JAULMES consists of a main loop containing two enumerations phases: A-Phase and B-Phase. The authors of [10] assume that the B-Phase controls the time complexity of this approach. Without going too far into the analysis of this technique, we found that the overall complexity is wrongly estimated. By considering a reasonable choices of parameters, it turns out that the time complexity of the algorithm is dominated most cases by the A-Phase.

Recall from [10],the analysis of JOUX-JAULMES algortihm:

The number of operations needed to execute the A-Phase is in:

$$\mathcal{C}_{A-Phase} = \mathcal{O}\Big(\max\big\{N_1 \log(N_2)\psi,\ (m+1-k)\psi \frac{n!}{(n-n_1-n_2)!p^{-k}}\big\} \Big).$$

While the B-phase requires:

$$\mathcal{C}_{B-Phase} = \mathcal{O}\Big(\max\big\{N_3 \log(N_4), (m+1-k)\psi \frac{n!^2}{(n-n_1-n_2)!(n-n_3-n_4)!} p^{-k}\big\} \Big),$$

Where, $N_i = \frac{n!}{(n-n_i)!}$. Hence, by considering the effect of the main loop which contains the 2 Phases, the total time complexity can be expressed as:

$$\phi \times \left(\mathcal{C}_{A-Phase} + \mathcal{C}_{B-Phase}\right),$$

Where $\phi = p^k$ is the cost of the main loop containing the two phases A and B. Experimental results show that for the set of parameter that minimize the total time complexity, the phase A dominates, and the total is much higher as announced. Therefore, the Joux-Jeaulmes algorithm is not efficient for solving PKP.

4 Complexity Analysis

Here, we try to estimate a lower bound of the complexity of algorithms solving PKP. As a simplification, we assume that the elementary operation is the computation of a tuple, namely a vector-matrix product in $\mathbb{F}_p$ or whatever should be more efficient to accomplish it, as for instance a result can be deduced from another one, where few changes have to be made. For the same reason, we assume also that the memory unit is the space needed to store a tuple.

In this section, we use the following notation: k-V is a tuple of k values, where k notes conveniently at the same time a subset of indexes and the number of elements of this subset. In the same spirit, $k.i$-A is a sub-matrix of A with the given subsets of indexes.

4.1 Simplest and most efficient algorithm

We recall here in Alg. 1 the first method for solving PKP. This methods exploits usefully the special form of the matrix $A = (A' \| I)$.

For this algorithm, time and space complexity can be approximated by these formulas :

$$\text{space} = \frac{n!}{(n-l)!} \tag{1}$$

$$\text{time} = \frac{n!}{(n-l)!} + \frac{n!}{(n-r)!} + \frac{n!}{p^k(n-(l+r))!} \tag{2}$$

In Fig. 1, the number m of equations is implicitly estimated as the closest integer of $\log(n!)/\log(p)$ and the extra equation from Georgiades is used.

4.2 Memory enhancement

Here we give in Alg. 2 a more synthetic description of Poupard's algorithm, and its detailed complexity. Notations are the ones in [5, Fig. 3]. It uses precomputed files and "Middle values" techniques.

Algorithm 1 A1 : Solve PKP (n,m,p)

Require: $0 \leq k \leq m$ and $l+r+m-k=n$

1: select k equations, and split their variables into two sets l and r
2: **for all** l-tuple l-V **do**
3: compute $C \leftarrow k.l\text{-}A \times l\text{-}V$
4: store efficiently $(C, l\text{-}V)$ in a file $F0$ so that given a value C, one can retrieve efficiently all the associated l-tuples
5: **end for**
6: **for all** r-tuple r-V **do**
7: compute $C \leftarrow -k.r\text{-}A \times r\text{-}V$
8: retrieve from $F0$ a list L of l-tuples associated with C
9: filter the list L keeping only the l-tuples compatible with r-V
10: **for all** l-tuple l-V in L **do**
11: compute the last variables s-$V \leftarrow (m-k).(l+r)\text{-}A \times (l+r)\text{-}V$
12: **if** the values s are compatible with $(l+r)$ **then**
13: $(l+r+s)$ is a solution
14: **end if**
15: **end for**
16: **end for**

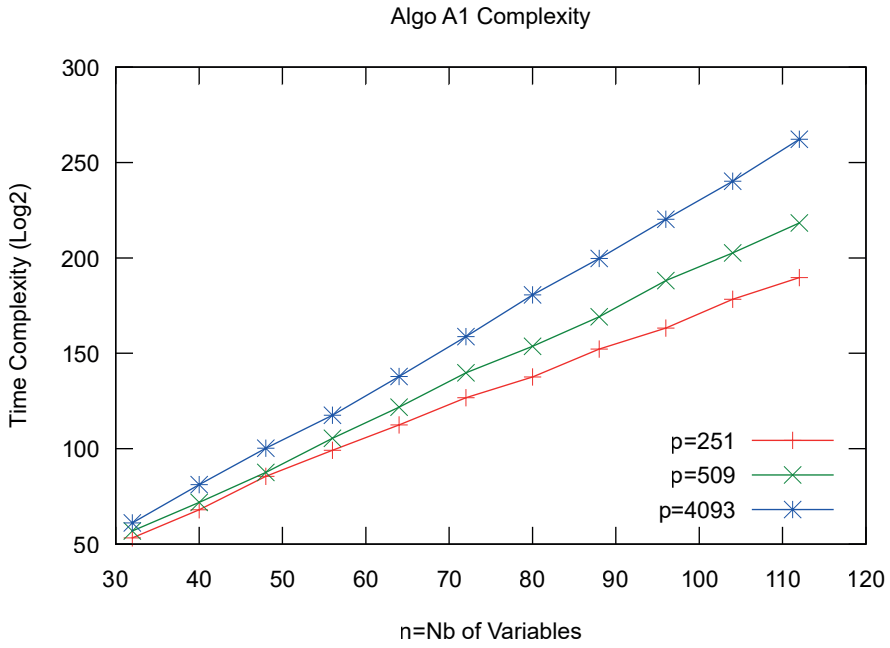

Fig. 1. Time complexity of Alg. 1 for various values of p

Algorithm 2 Poupard : Solve PKP (n,m,p)

Require: $i+j+r+m=n$ and $c+c'\leq m$ and $d\leq c$ and $l+k=r+d$

for all j-tuple j-V **do**
  compute $C0 \leftarrow c.j\text{-}A \times j\text{-}V$
  store efficiently $(C0, j\text{-}V)$ in a file $F0$ so that given a value $C0$, one can retrieve efficiently all the associated j-tuples
end for
for all l-tuple l-V **do**
  compute $C2 \leftarrow d.l\text{-}A \times l\text{-}V$
  store efficiently $(C2, l\text{-}V)$ in a file $F2$ so that given a value $C2$, one can retrieve efficiently all the associated l-tuples
end for
for all c-tuple of "Middle values" c-C **do**
  for all i-tuple i-V **do**
    compute $C0 \leftarrow c\text{-}C - c.i\text{-}A \times i\text{-}V$
    retrieve from $F0$ a list L of j-tuples associated with $C0$
    filter the list $L0$ keeping only the j-tuples compatible with i-V and get $(j+i)$-tuples
    for all $(j+i)$-tuple $j+i$-V of $L0$ **do**
      compute $C1 \leftarrow c'.(j+i)\text{-}A \times (j+i)\text{-}V$
      store efficiently $(C1, (j+i)\text{-}V)$ in a file $F1$ so that given a value $C1$, one can retrieve efficiently all the associated $(j+i)$-tuples
    end for
    for all k-tuple k-V **do**
      compute $C2 \leftarrow d\text{-}C - d.k\text{-}A \times k\text{-}V$
      retrieve from $F2$ a list $L2$ of l-tuples associated with $C2$
      filter the list $L2$ keeping only the l-tuples compatible with k-$V$and get $(l+k)$-tuples
      for all $(l+k)$-tuple $(l+k)$-V of $L2$ **do**
        compute $(c-d)\text{-}V \leftarrow (c-d)\text{-}C - (c-d).(l+k)\text{-}A \times (l+k)\text{-}V$ and keep only values compatible with $(l+k)$-V and get $(r+c)$-tuples
        for all c'-tuple c'-V compatible with $(r+c)$-V **do**
          compute $C1 \leftarrow -(c').(r+c)\text{-}A \times (r+c)\text{-}V$
          retrieve from $F1$ a list $L1$ of $j+i$-tuples associated with $C1$
          filter the list $L1$ keeping only the $(j+i)$-tuples compatible with $r+c+c'$-V and get $(j+i+r+c+c')$-tuples
          for all $(j+i+r+c+c')$-tuple $(j+i+r+c+c')$-V in $L1$ **do**
            compute the last variables s-$V \leftarrow (m-c-c').(j+i+r+c+c')\text{-}A \times (j+i+r+c+c')\text{-}V$
            if the values s are compatible with $(j+i+r+c+c')$ **then**
              $(j+i+r+c+c'+s)$ is a solution
            end if
          end for
        end for
      end for
    end for
  end for
end for

For this algorithm, time and space complexity can be approximated by these formulas :

$$\text{space} = \frac{n!}{(n-j)!} + \frac{n!}{(n-l)!} + \frac{n!}{p^c(n-(i+j))!} \tag{3}$$

$$\begin{aligned}\text{time} = {} & \frac{n!}{(n-j)!} + \frac{n!}{(n-l)!} + \frac{n!}{(n-(i+j))!} + \\ & \frac{p^c n!}{(n-k)!} + \frac{p^{c-d} n!}{(n-(k+l))!} + \frac{p^{c-d}(n-(c-d))!}{(n-(r+c+c'))!} \\ & + \frac{(n-(c-d))!}{p^d(n-(i+j+r+c+c'))!}\end{aligned} \tag{4}$$

As said in [5], the interest of this algorithm is for realistic attacks, where memory is limited. However from a theoretical point of view, the previous algorithm is the most efficient.

4.3 Probabilistic method

We here discuss the method named "Pre-computation on the A matrix" of [9]. Like previous methods, this new method aims at decreasing the complexity. Its specificity is to search for subsets of equations with fewer variables than expected. We first give an estimation of the probability that among a given set of m random equations in n variables over $\mathbb{F}_p$, there exists a subset of k equations in only r variables. Such a probability was given in [9], but here we explain how to compute it more accurately. Lets start with the following results.

Claim. The probability that a random linear equation in n variables over $\mathbb{F}_p$ is indeed in r variables is $\binom{n}{r}(\frac{1}{p})^{n-r}(\frac{p-1}{p})^r$.

Proof. Let a random linear equation in n variables over $\mathbb{F}_p$. Each of its coefficients is assumed to be randomly and uniformly distributed if $\mathbb{F}_p$. Therefore the probability that a given variable does not occur in the equation is the probability that its coefficient is 0, or so $\frac{1}{p}$. All the coefficients are assumed to be independently drawn at random. Therefore, the number of variables that appears in the equation follows the binomial distribution with parameters n and $\frac{1}{p}$; hence the result.

Claim. The probability that a set of k random linear equations in n variables over $\mathbb{F}_p$ is indeed in r variables is $\binom{n}{r}(\frac{1}{p^k})^{n-r}(1-\frac{1}{p^k})^r$.

Proof. The probability that a given variable does not occur in the k equations $\frac{1}{p^k}$. Therefore, the number of variables that appears in the k equations follows the binomial distribution with parameters n and $\frac{1}{p^k}$; hence the result.

Claim. In a linear space of m linear equations over $\mathbb{F}_p$, the number of distinct linear subspaces of k equations is

$$\prod_{i=0}^{k-1} \frac{p^m - p^i}{p^k - p^i}$$

Proof. The number of k-tuples of linearly independent equations is given by :

$$\prod_{i=0}^{k-1} p^m - p^i.$$

All the tuples that are equivalent bases of the same subspace are related by a given isomorphism over $\mathbb{F}_p^k$. These isomorphisms amount to $\prod_{i=0}^{k-1} p^k - p^i$. Hence the result.

Claim. In a linear space of m linear equations over $\mathbb{F}_p$, the number of distinct linear subspaces of k equations that can be expressed in r variables is approx.

$$\binom{n}{r} (\frac{1}{p^k})^{n-r} (1 - \frac{1}{p^k})^r \prod_{i=0}^{k-1} \frac{p^m - p^i}{p^k - p^i}.$$

Proof. Although the subspaces of dimension k in the linear space are not uniformly distributed over all possibilities, a good approximation is the product of the number of subspaces by the probability that one subspace has the required property. This result has been experimentally verified.

Although the method of using subspaces of equations with less variables speeds indeed the searching algorithm, it appears that the probability of finding such subsets is overwhelming small. Example: For PKP(64,37,251), the optimum value for k is 19. Therefore, finding k equations expressed in $r = n - m + k = 45$ is easy. The expected number of subspaces expressed in only 44 variables is $\approx 2^{-98.5}$.

5 Conclusion

The main thing that we have essentially looked at is the complexity analysis of the Permuted Kernel Problem. We have seen the most well-known methods for solving PKP. We presented briefly each one and updated some results that were not accurate or genuine, namely in [5,10]. Also, we discussed the approach of PATARIN-CHAUVAUD introduced in [9], which is the most efficient one and gave an explicit complexity formula for generic parameters.

References

1. Shamir, A. (1989, August). An efficient identification scheme based on permuted kernels. In Conference on the Theory and Application of Cryptology (pp. 606-609). Springer, New York, NY.
2. Gary, M., Johnson, D. (1979). Computers and Intractability: A Guide to NP-Completeness. New York: W H.
3. Fiat, A., Shamir, A. (1986, August). How to prove yourself: Practical solutions to identification and signature problems. In Advances in Cryptology—CRYPTO'86 (pp. 186-194). Springer, Berlin, Heidelberg.
4. Bennett, C. H., Bernstein, E., Brassard, G., Vazirani, U. (1997). Strengths and weaknesses of quantum computing. SIAM journal on Computing, 26(5), 1510-1523.

5. Poupard, G., (1997). A realistic security analysis of identification schemes based on combinatorial problems. European Transactions on Telecommunications.
6. Lyubashevsky, V. (2009, December). Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In International Conference on the Theory and Application of Cryptology and Information Security (pp. 598-616). Springer, Berlin, Heidelberg.
7. Georgiades, J. (1992). Some remarks on the security of the identification scheme based on permuted kernels. Journal of Cryptology, 5(2), 133-137.
8. Baritaud, T., Campana, M., Chauvaud, P., Gilbert, H. On the security of the permuted kernel identification scheme. In Annual International Cryptology Conference (pp. 305-311). (1992, August), Springer, Berlin, Heidelberg.
9. Patarin, J., Chauvaud, P. Improved algorithms for the permuted kernel problem. In Annual International Cryptology Conference (pp. 391-402). (1993, August) Springer, Berlin, Heidelberg.
10. Jaulmes, É., Joux, A. Cryptanalysis of pkp: a new approach. In International Workshop on Public Key Cryptography (pp. 165-172). (2001, February) Springer, Berlin, Heidelberg.
11. Joux, A., Lercier, R. "Chinese and Match", an alternative to Atkin's "Match and Sort" method used in the SEA algorithm. Mathematics of computation, 70(234), 827-836.
12. Joux, A., Lercier, R. Chinese and Match, an alternative to Atkin's Match and Sort method used in the SEA algorithm. (1999). Preprint.