On the Security of the Permuted Kernel Identification Scheme

Thierry BARITAUD Mireille CAMPANA Pascal CHAUVAUD Henri GILBERT

Centre National d'Etudes des Télécommunications
PAA/TSA/SRC
38-40 Rue du Général Leclerc
92131 Issy les Moulineaux
FRANCE

Abstract

A zero-knowledge identification scheme built upon the so-called Permuted Kernel Problem (PKP) was proposed by Adi Shamir in 1989 [1].
In this paper, we present a time-memory trade-off leading to a reduction of the computation time for solving the PKP problem, as compared with the best known attack mentioned in [1].

1. Introduction

In 1989, Adi Shamir proposed a new identification scheme, based on the intractability of the Permuted Kernel Problem (PKP)[1]. This scheme requires quite limited time, memory and communication resources, and is well suited for smart card implementation.

This paper investigates the security of the new scheme for some of the parameter values suggested in [1] as possible choices, subject to a further analysis of their security. Our main conclusion is that the smallest parameter values mentioned in [1] (n=32; m=16; p=251) are not recommended, at least for applications with strong security requirements. As a matter of fact, we show that for these values there exists a very simple time-memory trade-off leading to a faster solution of the PKP problem than the best known attack mentioned in [1]. For larger parameter values, this time-memory trade-off does not endanger the practical security of the PKP scheme, while the time, space and communication complexities of the PKP scheme stay within acceptable limits.

E.F. Brickell (Ed.): Advances in Cryptology - CRYPTO '92, LNCS 740, pp. 305-311, 1993.

2. The Permuted Kernel Problem

We are using, as far as possible, the notations of [1].

The PKP problem is the following :

Given :

a prime number p;
a $m \times n$ matrix $A = (a_{ij})_{i=1..m;j=1..n}$ over Z/pZ;
a n-vector $V = (V_j)_{j=1..n}$ over Z/pZ

Find : a permutation π over $\{1,..,n\}$ such that $A.V_\pi = 0$, where $V_\pi = (V_{\pi(j)})_{j=1..n}$.

In the PKP identification scheme, each prover uses a public instance (p,A,V) of the above problem. The values p, A, V are generated as follows :

- p and A are fixed values agreed by the users (and can be used by several provers). We will here assume that A is of rank m and is generated under the form [A',I], where A' is a fixed $m \times (n-m)$ matrix and I is the $m \times m$ identity matrix. (As mentioned in [1], this is not restrictive because both the prover and the verifier can apply Gaussian elimination to any $m \times n$ matrix without changing the kernel).

- V (the public key of a prover) is generated from a random permutation π and a random vector of the kernel of A (which serve as his secret key) in such a way that $A.V_\pi = 0$.

To convince a verifier of his identity, a prover gives him evidence of his knowledge of a solution π to the (p,A,V) instance by using a zero-knowledge protocol. The detail of this protocol, which is the main subject of [1], is outside the scope of this paper, since we are merely interested in the computational difficulty for an attacker of solving the (p,A,V) instance.

3. A time-memory trade-off for the PKP problem

Let (p, A, V) be an instance of the PKP problem generated as explained in Section 2. We are trying to find a permutation π such that $A.V_\pi = 0$.

Using the notations introduced in Section 2, we can rewrite $A.V_\pi = 0$ as :

$$\begin{pmatrix} a'_{1,1} & \cdots & a'_{1,n-m} & 1 & & \\ \vdots & & \vdots & & & \\ a'_{m,1} & \cdots & a'_{m,n-m} & & & 1 \end{pmatrix} \begin{pmatrix} V\pi(1) \\ \cdot \\ \cdot \\ \cdot \\ V\pi(n) \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}$$

We denote by (1) to (m) the relations provided by the rows 1 to *m* of the above matrix.

Note : in the sequel we are sometimes using the notation $A=(a_{ij})_{i=1..m;j=1..n}$ instead of the above 'reduced' notation $A=[A',I]$; $A'=(a'_{ij})_{i=1..m;j=1..n-m}$ to denote the elements of the A matrix.

We first try to solve the equations (1) to (k), where k is a parameter of our algorithm ($0 \le k \le m$). Because of the structure of the A matrix, these k relations involve only the n-m+k unknown values $V_{\pi(1)}, \ldots, V_{\pi(n-m+k)}$.

We introduce an additional parameter k' (such that $0 \le k' \le n-m+k$), which determines the amount of storage to be performed in the precomputation phase.

There are two main steps in the proposed searching method :

Step1 : precomputation

For each of the $\frac{n!}{(n-k')!}$ possible values for the $(V_{\pi(1)}, \ldots, V_{\pi(k')})$ k'-uple, we calculate the corresponding contributions :

$$b_1 = \sum_{j=1}^{k'} a_{1,j}.V\pi(j) \ ;$$

$$\cdots$$

$$b_k = \sum_{j=1}^{k'} a_{k,j}.V\pi(j) \ .$$

to the relations (1) to (k).

We store these values and the obtained results $b_1, \ldots, b_k$ in such a way that for each of the p^k possible $(b_1, \ldots, b_k)$ values, the list of the corresponding $(V_{\pi(1)}, \ldots, V_{\pi(k')})$ k'-uples can be accessed in very few elementary operations.

The cost of this precomputation step is $\frac{n!}{(n-k')!}$ matrix-vector products. The storage required is about $\frac{n!}{(n-k')!}$ k'-uples. The average number of k'-uples corresponding to a $(b_1, ...,b_k)$ value is $\frac{n!}{(n-k')!}\, p^{-k}$.

We also introduce the convention that k'=0 means : no precomputation.

Step 2 : exhaustive trial

We perform an exhaustive trial of the $\frac{n!}{(m+k'-k)!}$ possible values for the $(V_{\pi(k'+1)}, ..., V_{\pi(n-m+k)})$ vector.

For each tried value, we calculate the corresponding contributions to the relations (1) to (k) :

$$c_1 = \sum_{j=k'+1}^{n-m+k} a_{1,j}.V\pi(j) ;$$

..

$$c_k = \sum_{j=k'+1}^{n-m+k} a_{k,j}.V\pi(j) .$$

We can now use the precomputations of Step 1 to obtain a list of possible $(V_{\pi(1)}, ..., V_{\pi(k')})$ k'-uples. As a matter of fact, the relations (1) to (k) can be rewritten :

$b_1+c_1 = 0;$

..

$b_k+c_k = 0.$

so that the $(V_{\pi(1)}, ..., V_{\pi(k')})$ k'-uple does necessary belong to the list of possible k'-uples for the $(-c_1,...,-c_k)$ value of $(b_1,...,b_k)$.

For each tried $(V_{\pi(k'+1)}, ..., V_{\pi(n-m+k)})$ vector, we obtain in average $\frac{n!}{(n-k')!}\, p^{-k}$ $(V_{\pi(1)}, ..., V_{\pi(k')})$ values. Some of them have to be discarded because some values of $\{V_{\pi(1)}, ..., V_{\pi(k')}\}$ are already contained in $\{V_{\pi(k'+1)}, ..., V_{\pi(n-m+k)}\}$.

For each remaining $(V_{\pi(1)}, ..., V_{\pi(n-m+k)})$ vector, the still unsolved relations (k+1) to (m) provide successively one single possible value for the numbers $V_{\pi(n-m+k+1)}$ to $V_{\pi(n)}$.

At each stage of this process of extending a $(V_{\pi(1)}, ..., V_{\pi(n-m+k)})$ candidate to a $(V_{\pi(1)}, ..., V_{\pi(n)})$ solution, it has to be checked that the obtained values are non repeating and belong to the $\{V_1, ..., V_n\}$ set. This can be done in very few elementary operations.

The procedure described above finds all the existing solutions (i.e. the secret vector $(V_{\pi(1)}, ..., V_{\pi(n)})$ and the other solutions if there are some). The required memory for this step is negligible. The required time is about $\text{Sup}\left(\frac{n!}{(m+k'-k)!}\frac{n!}{(n-k')!}p^{-k}, \frac{n!}{(m+k'-k)!}\right)$ matrix vector products.

In Summary, we have shown that for each (k,k') pair ($0 \le k \le m$; $0 \le k' \le n-m+k$) an instance (p,A,V) of the PKP problem can be solved in time :

$$\frac{n!}{(n-k')!} + \text{Sup}\left(\frac{n!}{(m+k'-k)!}\frac{n!}{(n-k')!}p^{-k}, \frac{n!}{(m+k'-k)!}\right) \text{ matrix-vector products} \quad \text{(i)}$$

(cf note below)

and space :

$$\frac{n!}{(n-k')!} \text{ k'-vectors} \quad \text{(ii)}.$$

Note : all the matrix vector products considered here involve a submatrix of A. Therefore, the cost of each such product can be reduced to very few elementary operations (mod p additions and accesses to arrays), at the expense of a marginal increase of the required space, by precomputing all the linear combinations modulo p of some subsets of the rows in A.

Discussion on the values of k and k'

The value k=0 corresponds to an exhaustive trial of all the $\frac{n!}{m!}$ possible $(V_{\pi(1)}, ..., V_{\pi(n-m)})$ values. The space cost is negligible and the time cost is about $\frac{n!}{m!}$. For larger values of k, precomputations on the k first equations may lead to an improved time cost, at the expense of increasing the required storage. Too large values of k are suboptimal, because the set of n-m+k variables involved in the equations (1) to (k) increases too much.

For a fixed value of k, the required amount of storage is an increasing function of k'; the time required is a decreasing function of k' as long as $k' \le (n-m+k)/2$ and $\frac{n!}{(n-k')!} \le p^k$ (the first condition says that the time spent on the precomputation should not exceed the time spent on step 2; the second condition says that the average number of k'-uples corresponding to a $(b_1, ..., b_k)$ value should be less than 1). Too large values of

k' (such that the two above conditions are not realised) are suboptimal, because they lead to an increased memory cost without reducing the time cost.

4. Impact on the practical security of the PKP scheme

Table 1 gives the time and memory costs, calculated in using (i) and (ii), when n=32; m=16; p=251. Only the k values in the [0..9] interval and the k' values in the [0..15] interval have been considered. The value k=0 leads to a time cost of about 2^{73}, which is very close to the 2^{76} complexity of the best attack mentioned in [1]. The most interesting values are obtained for k=5; k'=8 (**time :2^{60} matrix vector products; memory 2^{38} k'-uples**) and k=6; k'=10 (**time 2^{56}; memory 2^{47}**).

The obtained complexity values are considerable, but the two above trade-offs cannot be regarded as strictly computationally infeasible, as it was the case for the attack mentioned in [1]. Therefore, the parameter values n=32; m=16; p=251 are not recommended for very secure applications.

Table 2 gives the time and memory costs corresponding to the parameter values n=64; m=37; p=251 for some of the k and k' parameter values. Some of the obtained time costs are substantially lower than the 2^{184} complexity of the best attack mentioned in [1]. For example, for k=8 and k'=11, the obtained time cost is 2^{137} matrix vector products, and the required storage is 2^{64} k'-uples. However, due to the very large values of the obtained time costs, the attacks summarised in Table 2 are computationally infeasible.

5. Concluding remarks

Independently of our work, the security of the PKP problem has also been investigated by J. Georgiades and the results are summarized in [2]. His method, which is based on the resolution of quadratic equations, is less efficient in time cost than the one described in this paper, but requires a negligible amount of storage to solve the PKP problem. We do not know whether both approaches can be combined efficiently.

References

[1] Adi Shamir, "An Efficient Identification Scheme Based on Permuted Kernels", Proceedings of Crypto'89, Springer Verlag.

[2] J. Georgiades, "Some Remarks on the Security of the Identification Scheme Based on the Permuted Kernel", Journal of Cryptology, Volume 5, Number 2, 1992.

k= / k'=	0	1	2	3	4	5	6	7	8	9	space(k')
0	73	77	81	85	88	92	95	99	102	105	0
1	74	73	77	81	85	88	92	95	99	102	5
2	75	71	73	77	81	85	88	92	95	99	9
3	75	72	69	73	77	81	85	88	92	95	14
4	76	72	68	69	73	77	81	85	88	92	19
5	76	73	69	65	69	73	77	81	85	88	24
6	77	73	69	66	65	69	73	77	81	85	29
7	77	73	70	66	63	65	69	73	77	81	33
8	77	73	70	66	63	60	65	69	73	77	38
9	77	73	70	67	63	59	60	65	69	73	43
10	77	73	70	67	63	60	56	60	65	69	47
11	76	73	70	66	63	60	56	56	60	65	52
12	76	73	69	66	63	60	57	56	57	60	56
13	75	72	69	66	63	61	60	60	60	60	60
14	75	72	69	66	65	65	65	65	65	65	65
15	74	71	69	69	69	69	69	69	69	69	69

Table 1 : base 2 log of time cost t and space cost s when n=32; m=16; p=251
(example : for k=6 and k'=10, t=2**56 and s=2**47)

k= / k'=	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	space
0	152	157	163	168	173	178	183	188	193	198	202	207	212	216	221	226	0
1	153	152	157	163	168	173	178	183	188	193	198	202	207	212	216	221	6
2	154	151	152	157	163	168	173	178	183	188	193	198	202	207	212	216	11
3	154	152	149	152	157	163	168	173	178	183	188	193	198	202	207	212	17
4	155	152	150	147	152	157	163	168	173	178	183	188	193	198	202	207	23
5	155	153	150	148	147	152	157	163	168	173	178	183	188	193	198	202	29
6	156	153	151	148	145	147	152	157	163	168	173	178	183	188	193	198	35
7	156	154	151	149	146	143	147	152	157	163	168	173	178	183	188	193	41
8	157	154	152	149	146	144	142	147	152	157	163	168	173	178	183	188	47
9	157	154	152	149	147	144	142	142	147	152	157	163	168	173	178	183	53
10	157	155	152	150	147	145	142	139	142	147	152	157	163	168	173	178	58
11	157	155	152	150	148	145	142	140	137	142	147	152	157	163	168	173	64
12	157	155	153	150	148	145	143	140	138	136	142	147	152	157	163	168	70
13	157	155	153	150	148	145	143	140	138	135	136	142	147	152	157	163	76
14	157	155	153	150	148	146	143	141	138	136	133	136	142	147	152	157	81
15	157	155	153	150	148	146	143	141	138	136	133	131	136	142	147	152	87
16	157	155	153	150	148	146	143	141	138	136	133	131	131	136	142	147	93
17	157	155	153	150	148	146	143	141	139	136	134	131	129	131	136	142	98
18	157	155	152	150	148	146	143	141	139	136	134	131	129	126	131	136	104
19	157	154	152	150	148	145	143	141	138	136	134	131	129	126	126	131	109
20	156	154	152	150	148	145	143	141	138	136	134	131	129	126	124	126	115
21	156	154	152	149	147	145	143	140	138	136	133	131	129	126	124	122	120
22	155	153	151	149	147	145	142	140	138	136	133	131	129	127	126	126	126
23	155	153	151	149	146	144	142	140	138	135	133	132	131	131	131	131	131
24	154	152	150	148	146	144	142	140	138	137	136	136	136	136	136	136	136
25	154	152	150	148	146	144	142	142	142	142	142	142	142	142	142	142	142
26	153	151	149	148	147	147	147	147	147	147	147	147	147	147	147	147	147
27	153	153	152	152	152	152	152	152	152	152	152	152	152	152	152	152	152

Table 2 : base 2 log of time cost t and space cost s when n=64; m=37; p=251
(example : for k=8 and k'=11 , t=2**137 and s=2**64)