Designs, Codes and Cryptography, ???, 1–28 (???)

A New $\mathcal{NP}$-Complete Problem and Public-Key Identification

DAVID POINTCHEVAL david.pointcheval@ens.fr
Département d'Informatique, École Normale Supérieure, Paris, France

GUILLAUME POUPARD guillaume.poupard@m4x.org
Direction Centrale de la Sécurité des Systèmes d'Information, Paris, France

Received December 1997 ; Revised July 2001

Abstract. The appearance of the theory of zero-knowledge, presented by Goldwasser, Micali and Rackoff in 1985, opened a way to secure identification schemes. The first application was the famous Fiat-Shamir scheme based on the problem of modular square roots extraction. In the following years, many other schemes have been proposed, some Fiat-Shamir extensions but also new discrete logarithm based schemes. Therefore, all of them were based on problems from number theory. Their main common drawback is high computational load because of arithmetical operations modulo large integers. Implementation on low-cost smart cards was made difficult and inefficient.

With the Permuted Kernels Problem (PKP), Shamir proposed the first efficient scheme allowing for an implementation on such low-cost smart cards, but very few others have afterwards been suggested.

In this paper, we present an efficient identification scheme based on a combinatorial $\mathcal{NP}$-complete problem: the Permuted Perceptrons Problem (PPP). This problem seems hard enough to be unsolvable even with very small parameters, and some recent cryptanalysis studies confirm that position. Furthermore, it admits efficient zero-knowledge proofs of knowledge and so it is well-suited for cryptographic purposes. An actual implementation completes the optimistic opinion about efficiency and practicability on low-cost smart cards, and namely with less than 2KB of EEPROM and just 100 Bytes of RAM and 6.4 KB of communication.

Keywords: Zero-Knowledge Identification, (Permuted) Perceptrons Problem, $\mathcal{NP}$-Complete Problem, Simulated Annealing

1. Introduction

An interactive identification protocol involves two entities, Alice and Bob. Alice tries to convince Bob that she is really Alice. She has a public key that everybody knows and an associated private key only known to her and that nobody else can compute. In order to identify herself, Alice proves that she knows this private key. Usually the public key is an instance of a difficult problem while the private key is one of its solutions.

The theory of zero-knowledge [18] shows that one can prove some knowledge in such a way that the verifier gets the conviction that the prover really knows without learning anything else. The first zero-knowledge protocols were based on number

theoretical problems (Fiat-Shamir [9] and its variants [19, 27, 28] which are based on modular roots [40], Schnorr [41] which is based on the discrete logarithm problem). Even if they had, more or less recently, numerous improvements [22, 4, 12, 14, 5, 15, 43, 35, 36, 37, 33], they have two major drawbacks, both linked to the underlying problems.

- The different problems (factorization, RSA and the discrete logarithm problem), which are used because of there strong algebraic properties, are related. Then a breakthrough would probably involve many, and possibly, all of them. As one can remark, for the moment, the same NFS technique provides the best algorithm against all of them.
- Modular operations using a large modulus, like multiplications or exponentiations, are hard to perform. Even if they are practical on smart cards using arithmetical co-processors, such devices are still expensive (at least twice or three times the price of the cheapest chips).

1.1. Related Work

In order to circumvent the heavy computational load of previous protocols based on number theoretical problems, new schemes have appeared since 1989. They rely on combinatorial $\mathcal{NP}$-complete problems and require only operations involving very small numbers: the Permuted Kernels Problem ($\mathcal{PKP}$) [42], the Syndrome Decoding ($\mathcal{SD}$) [44, 46] or Constrained Linear Equations ($\mathcal{CLE}$) [45]. But this list is almost exhaustive.

1.2. Achievement

This paper investigates the so-called "Perceptrons Problem", an $\mathcal{NP}$-complete problem which comes from learning machines. It has been introduced in cryptography by the first author [31, 32]. Then, Knudsen and Meier improved the analysis of the problem difficulty.

The statement of Perceptions Problem is very simple: given m vectors X^i with coordinates equal to $+1$ or -1 ($X^i \in \{-1, +1\}^n$) as constraints, one wants to find a vector $V \in \{-1, +1\}^n$ such that all the products $X^i \cdot V$, for $i = 1, \ldots, m$, are nonnegative.

1.3. Outline of the Paper

After a more precise description of this problem and some variants, namely the Permuted Perceptrons Problem, we study their properties in the complexity theoretic setting. Next, we provide a more practical-oriented analysis of their difficulty in order to evaluate the size of the parameters required for cryptographic purpose. Then, we present some zero-knowledge interactive identification protocols based on the Permuted Perceptrons Problem, with an evaluation of their security. Fi-

nally we show that those protocols are well-suited for smart card applications: an implementation has been realized and we present the performance.

2. The Perceptrons Problems

This section is devoted to formally present the main problems from a complexity theoretic point of view. We first study the Perceptrons Problem. Then, we focus on a sub-case, the Permuted Perceptrons Problem.

2.1. The Perceptrons Problem

Let us first present this new problem. Then, we state several properties in the complexity theory setting: this problem is $\mathcal{NP}$-complete, and its optimization variation is MAX-$\mathcal{SNP}$-hard.

We call the following problem the *Perceptrons Problem* (or $\mathcal{PP}$) because of its similarities with the "Ising Perceptron Problem" in learning theory. This latter problem consists in learning a half-space given some samples in the N-dimensional unit ball classified according to whether they are in the half-space or not. The designation "Ising" refers to the ± 1 constraint, which is present in the original Ising model of magnetism with N interacting spins.

In the following, we call an *ε-matrix* (resp. an *ε-vector*) a matrix (resp. a vector) which components are either $+1$ or -1. Furthermore, all the products between matrices and vectors are made in the ring of integers, denoted $\mathbb{Z}$.

Problem 1 (The Perceptrons Problem – $\mathcal{PP}$)
Given: *an ε-matrix A of size $m \times n$.*
Question: *is there an ε-vector Y such that $A \cdot Y \geq 0$?*

THEOREM 1 *The Perceptrons Problem is $\mathcal{NP}$-complete.*

Proof: As usual, this proof needs two steps: first this problem lies in $\mathcal{NP}$ and it is furthermore $\mathcal{NP}$-hard.

$\mathcal{PP} \in \mathcal{NP}$. First, this problem is clearly in $\mathcal{NP}$. A witness is an n-dimensional ε-vector, and its correctness can be checked, by a simple matrix multiplication, within time $\mathcal{O}(mn)$.

$\mathcal{PP}$ is $\mathcal{NP}$-hard. In order to prove that this problem is $\mathcal{NP}$-hard, we polynomially reduce an instance $\mathcal{C} = \{C^1, \ldots, C^q\}$ of 3-$\mathcal{SAT}$ into an instance of $\mathcal{PP}$.

The intuition behind the reduction relies on a specific coding of clauses and truth assignments, both as vectors. It is made in such a way that an unsatisfied clause provides a smaller dot product, between its coding vector and the truth assignment coding, than a satisfied clause. More precisely, it provides -6 whereas a satisfied one provides -2, 2 or +6. In a second step, we add some constraints to make a $\mathcal{PP}$ solution vector to be a valid coding of a truth assignment. Finally, we add some components to increase the dot products from -2 to 0.

Initialization: A 3-$\mathcal{SAT}$ instance $\mathcal{C}$, with parameters (q, k), consists of q 3-clauses: for any i, the clause C^i is the disjunction of three distinct literals $\{\ell_1^i, \ell_2^i, \ell_3^i\}$ over the k boolean variables $x_1, \ldots, x_k$. More concretely, for any j, $\ell_j^i \in \{x_1, \bar{x}_1, \ldots, x_k, \bar{x}_k\}$.

First Step: Each C^i is first encoded by an ε-vector X^i of size $2k$ which components depend on the presence or absence of each literal: for any $p \in \{1, \ldots, k\}$,

$$X_p^i = X_{p+k}^i = \begin{cases} +1 & \text{if } x_p \in C^i, \\ -1 & \text{if } \bar{x}_p \in C^i, \end{cases} \qquad X_p^i = -X_{p+k}^i = +1 \quad \text{otherwise.}$$

In the same way, any truth assignment D is also encoded by an ε-vector V of size $2k$: for any $p \in \{1, \ldots, k\}$,

$$V_p = V_{p+k} = \begin{cases} +1 & \text{if } x_p \in D, \\ -1 & \text{if } \bar{x}_p \in D. \end{cases}$$

Therefore, with both encodings, one can remark that variables x_p which are not in C^i do not participate to the dot product, since whatever the related truth value, it leads to either $+1 - 1 = 0$ or $-1 + 1 = 0$. On the contrary, a satisfied literal in C^i participates with $+2$, whereas an unsatisfied literal participates with -2. As a consequence, since a clause is satisfied if at least one of its literals is true:

$$\begin{aligned} C^i \text{ is satisfied under } D &\iff X^i \cdot V \in \{-2, +2, +6\}, \\ C^i \text{ is not satisfied under } D &\iff X^i \cdot V = -6. \end{aligned}$$

Second Step: However, an ε-vector V actually encodes a truth assignment if $V_p = V_{p+k}$ for any $p \in \{1, \ldots, k\}$. Let us add further constraints with the k following ε-vectors $Z^1, \ldots, Z^k$: for any $j \in \{1, \ldots, k\}$ and for any $p \in \{1, \ldots, k\}$,

$$Z_p^j = -Z_{p+k}^j = \begin{cases} +1 & \text{if } p = j, \\ -1 & \text{if } p \neq j. \end{cases}$$

An ε-vector V actually encodes a truth assignment if and only if $Z^j \cdot V = 0$ for any $j \in \{1, \ldots, k\}$. Therefore, a solution V to the following system

$$\begin{cases} X^i \cdot V \ \geq \ -2 & \text{for } i = 1, \ldots, q, \\ Z^j \cdot V \ = \ 0 & \text{for } j = 1, \ldots, k. \end{cases}$$

encodes a valid truth assignment which satisfies all the clauses C^i, for $i = 1, \ldots, q$.

Final Step: In order to make the inequalities relative to zero, we add two components to each vector: we extend every X^i in $\tilde{X}^i$ with two more components equal to $+1$, and we force the new components of V to be $+1$ extending every Z^j into $\tilde{Z}_+^j$ with $(+1, -1)$ and $\tilde{Z}_-^j$ with $(-1, +1)$:

$$\begin{aligned} \tilde{X}^i &= (X^i, +1, +1), \quad \text{for any } i \in \{1, \ldots, q\}, \\ \tilde{Z}_+^j &= (Z^j, +1, -1) \quad \text{and } \tilde{Z}_-^j = (Z^j, -1, +1), \quad \text{for any } j \in \{1, \ldots, k\}. \end{aligned}$$

Therefore, an ε-vector $\tilde{V}$ of size $2k+2$ which satisfies

$$\begin{cases} \tilde{X}^j \cdot \tilde{V} &= X^j \cdot V + \tilde{V}_{2k+1} + \tilde{V}_{2k+2} \geq 0 \text{ for } j = 1, \ldots, q, \\ \tilde{Z}^j_+ \cdot \tilde{V} &= Z^j \cdot V + \tilde{V}_{2k+1} - \tilde{V}_{2k+2} = 0 \text{ for } j = 1, \ldots, k, \\ \tilde{Z}^j_- \cdot \tilde{V} &= Z^j \cdot V - \tilde{V}_{2k+1} + \tilde{V}_{2k+2} = 0 \text{ for } j = 1, \ldots, k, \end{cases}$$

furthermore satisfies $Z^j \cdot V = 0$ for $j = 1, \ldots, k$ and thus $\tilde{V}_{2k+1} = \tilde{V}_{2k+2}$ which thus leads to $X^j \cdot V \geq -2$ for $i = 1, \ldots, q$.

Then, we have polynomially transformed an instance of 3-$\mathcal{SAT}$ with q clauses over k variables, into an instance of $\mathcal{PP}$ of size $m \times n$ with $m = q + 4k$ and $n = 2k + 2$. ■

The problem $\mathcal{PP}$ is difficult to solve in the worst case, but what about its approximation? We now claim that it cannot be efficiently approximated either. First, let us formally define the optimization problem.

Problem 2 (Optimization Problem – MAX-$\mathcal{PP}$**)**
Given: *an ε-matrix A of size $m \times n$.*
Question: *find an ε-vector Y such that the number of nonnegative components of the vector $A \cdot Y$ is maximal.*

Using notations introduced by Papadimitriou and Yannakakis [29], it is clear that this problem can be approximated within a factor 2. In fact, for a given instance A, let us denote by k the maximal number of components of a product $A \cdot Y$ that we can make simultaneously nonnegative with a well-chosen Y. Considering a random vector and its opposite, we get one for which the product by A admits more than $n/2$ nonnegative components, which is greater than $k/2$, since $k \leq n$.

Nevertheless, from the following theorem, there exists a constant $\varepsilon > 0$ for which one cannot approximate this problem within a factor $1 + \varepsilon$, unless $\mathcal{P} = \mathcal{NP}$.

THEOREM 2 MAX-$\mathcal{PP}$ *is* MAX-$\mathcal{SNP}$*-hard.*

Proof: The proof is similar to the previous one (for Theorem 1), but we provide an $\mathcal{L}$*-reduction* [29] from the canonical MAX-$\mathcal{SNP}$-complete problem: MAX-2-$\mathcal{SAT}$. An $\mathcal{L}$-reduction is a polynomial reduction from the original problem $\mathcal{P}_1$ into the other problem $\mathcal{P}_2$, for which there also exists a polynomial algorithm which transforms any solution of $\mathcal{P}_2$ into a solution of $\mathcal{P}_1$ with a linear ratio. Therefore, if one can linearly approximate $\mathcal{P}_2$ (within a factor $1 + \varepsilon$), one can also linearly approximate $\mathcal{P}_1$.

Basically, we encode the same way as before, a 2-clause, but this time, a satisfied 2-clause provides a nonnegative dot product between its coding vector and the truth assignment coding. Then, we add many constraints (which can be satisfied altogether) to enforce the optimal solution to be a valid truth assignment coding.

With so many easily satisfied 2-clauses, one can show that any good approximation (within a factor $1 + \varepsilon$) for the obtained MAX-$\mathcal{PP}$ instance can be proven to satisfy all the additional constraints, then it encodes a truth assignment. Each other satisfied inequality exactly corresponds with a satisfied clause. ■

3. The Permuted Perceptrons Problem

As we have just seen, the Perceptrons Problem is $\mathcal{NP}$-complete. Consequently, according to a very general result [17], it admits a zero-knowledge interactive proof of knowledge. This latter could be turned into a zero-knowledge identification protocol. Nevertheless, in order to get an efficient protocol, we define a variant of this problem: the Permuted Perceptrons Problem (or $\mathcal{PPP}$). For this new problem, we need the notion of multi-sets of size m denoted by $\{\{a_1, \ldots, a_m\}\}$. The multi-sets are simply sets which may contain some repeated elements.

Problem 3 (The Permuted Perceptrons Problem – $\mathcal{PPP}$)
Given: *an ε-matrix A of size $m \times n$ and a multi-set S of m nonnegative integers.*
Question: *is there an ε-vector Y such that the multi-set S is equal to $\{\{(A \cdot Y)_i | i = 1, \ldots m\}\}$?*

THEOREM 3 *The Permuted Perceptrons Problem is $\mathcal{NP}$-complete.*

Proof: For this proof, we reduce an instance of the $\mathcal{NP}$-complete problem ONE-IN-THREE 3-$\mathcal{SAT}$ [10] into an instance of $\mathcal{PPP}$. The ONE-IN-THREE 3-$\mathcal{SAT}$ problem consists in solving a 3-$\mathcal{SAT}$ instance in such a way that in each 3-clause, exactly one literal is true.

More precisely, the obtained $\mathcal{PPP}$ instance relies in particular sub-group of instances, namely where $S = \{\{0, \ldots, 0\}\}$. Indeed, using exactly the same reduction as for Theorem 1, satisfied clauses in the one-in-three sense lead to the equation $X^i \cdot V = -2$ and the system to solve becomes:

$$\begin{cases} X^i \cdot V &= -2 \quad \text{for } i = 1, \ldots, q, \\ Z^j \cdot V &= 0 \quad \text{for } j = 1, \ldots, k. \end{cases}$$

which makes a solution V to encode a valid truth assignment which satisfies all the clauses C^i in the one-in-three sense, for $i = 1, \ldots, q$. Adding the same two-components, we make a system with only equalities with 0.

Since a sub-problem of $\mathcal{PPP}$ is $\mathcal{NP}$-complete, the general $\mathcal{PPP}$ problem is also $\mathcal{NP}$-complete. ■

4. Security of the Problem

In this section, we analyze the practical security of the two problems we have just presented. Since they have never been used in computer science for cryptographic purposes, a careful analysis of their complexity is necessary. Notice that the $\mathcal{NP}$-complete property does not guarantee that those problems are really well fitted for cryptographic applications because two additional properties also have to be verified.

- First, the size of the parameters that one has to use in order to prevent known attacks must not lead to inefficiency in terms of computation load or amount of transmission.

- Furthermore, the $\mathcal{NP}$-completeness only guarantees the existence of worst cases which solutions are hard to compute, but we need a much stronger property: all, but at most a negligible part of the instances, must be hard to solve.

In the following, we only focus on odd values for m and n, for technical reasons. We note $m = 2q + 1$ and $n = 2p + 1$. First, we study the link between m and n in order to have, on average, only one solution per instance of $\mathcal{PPP}$. Then we review several attacks against $\mathcal{PP}$ and we extend them to $\mathcal{PPP}$. Finally, we propose practical sizes for cryptographic purpose.

4.1. Number of Solutions for $\mathcal{PP}$ and for $\mathcal{PPP}$

As we will see in the following, the choice of the parameters m and n depends on the efficiency of the attacks. But we can first try to find a relation between them to maximize the complexity of any attack. On the one hand, if m is too large the system will be too constrained. In fact, let us consider the "Ising Perceptron Problem" where a vector $V \in \{+1, -1\}^n$ is given, it defines the half-space $V \cdot X \geq 0$ denoted H_V, as well as a number m of samples $X_1, \ldots, X_m$ in the n-dimensional unit ball classified according to whether they are in H_V or not. Baum, Boneh and Garrett [2] proved that their genetic algorithm can find V for $m = \mathcal{O}(n)$. On the other hand, if m is too small there will be a lot of solutions. In both cases, most of the attacks will be more efficient, so we suggest to take m and n such that $\mathcal{PPP}$ admits, on average, just one solution. To do so, we need to know the average number of solutions for $\mathcal{PP}$ when we know there is at least one and the probability to get a given multi-set S.

In order to evaluate the expected number of solutions, we have to make precise the probability distribution followed by the inputs A and V. In the sequel, we use the natural distribution which comes from the following construction:

- one uniformly chooses random ε-vector V and ε-matrix A.
- for every row A_i of A, if $A_i \cdot V < 0$ then one replaces this row by its opposite.

Now, we evaluate, by a combinatorial method, the total number of solutions for such an instance of $\mathcal{PP}$ which admits at least one solution: the private key. Let $\alpha \in \{-n, \ldots, n\}$ and X, V be two ε-vectors such that $X \cdot V = \alpha$. Let us use d_H to denote the Hamming distance between two vectors (the number of distinct coordinates), and $\#S$ to denote the cardinality of a given set S. For any integers k and δ, and for any ε-vector W such that $d_H(W, V) = k$ and $X \cdot W = \delta$, we have

$$\begin{aligned} X \cdot W &= \#\{i \,|\, W_i = X_i\} - \#\{i \,|\, W_i \neq X_i\} = \delta \\ &\left(\frac{n+\alpha}{2} - \beta + \gamma\right) - \left(\frac{n-\alpha}{2} - \gamma + \beta\right) = \alpha + 2\gamma - 2\beta, \end{aligned}$$

where every parameters are defined as described on the figure 1, and namely with the relation $k = \beta + \gamma$. Therefore, one gets

$$\beta(k, \alpha, \delta) = \frac{k}{2} - \frac{\delta - \alpha}{4} \text{ and } \gamma(k, \alpha, \delta) = \frac{k}{2} + \frac{\delta - \alpha}{4}$$

with the constraints $0 \leq \beta, \gamma \leq k$, $0 \leq \beta \leq (n+\alpha)/2$ and $0 \leq \gamma \leq (n-\alpha)/2$, Then, δ lies in the set

$$E(n,k,\alpha) = \left\{ \begin{array}{c} \delta - \alpha \in 4\mathbb{Z} \\ -2 \cdot \min\{k, n-k+\alpha\} \leq \delta - \alpha \leq 2 \cdot \min\{k, n-k-\alpha\}. \end{array} \right\}.$$

Let us consider the ε-vectors W at an even distance k from V (*i.e.* $k \in 2\mathbb{N}$):

$$\Pr_{\substack{W \\ d_H(V,W)=k}}[X \cdot W = \delta | X \cdot V = \alpha] = \frac{f(n,k,\alpha,\delta)}{\binom{n}{k}},$$

$$\text{where } f(n,k,\alpha,\delta) = \binom{\frac{n+\alpha}{2}}{\beta(k,\alpha,\delta)} \binom{\frac{n-\alpha}{2}}{\gamma(k,\alpha,\delta)}.$$

By summing over every nonnegative δ, we get the probability for W to provide a nonnegative dot product with V. Thus, for any vector Y, the number of solutions at even distance k from V, such that $A \cdot V = Y$, is equal to

$$N(m,n,k,Y) = \prod_{i=1}^{m} \sum_{\substack{\delta \in E(n,k,Y_i) \\ \delta \geq 0}} \frac{f(n,k,Y_i,\delta)}{\binom{n}{k}}.$$

Then, by summing over every possible even distance k, we get the number of solutions at even distance from V such that $A \cdot V = Y$, denoted by $N(m,n,Y)$.

Consequently, we can evaluate the average number of solutions $N(m,n)$ for an instance of size $m \times n$, when the known solution vector $Y = A \cdot V$ follows a normal distribution, which will be the case in practice, *i.e.* $p_{n,\alpha} = \Pr[Y_i = \alpha] \approx e^{-\frac{\alpha^2}{2n}} \sqrt{8/\pi n}$ if α is even, and 0 otherwise.

Let S^m be a multiset with m nonnegative integers. We will use the following notations:

Notation 1 $P_{m,n,S^m} = \Pr_Y[\{\{(A \cdot Y)_i\}\} = S^m \mid A \cdot Y \geq 0]$.

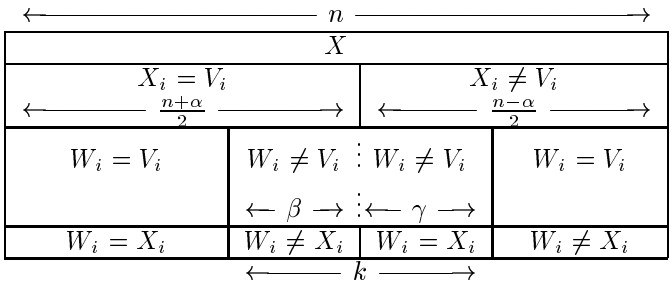

Figure 1. Number of Solutions of $\mathcal{PP}$

Notation 2 *$|S^m|_j$ represents the number of elements of S^m equal to j.*

If the rank of A is maximal, *i.e.* there exists an m-size submatrix R of A with an invertible determinant in $\mathbb{Z}$, then

$$P_{m,n,S^m} = m! \prod_{j=1}^{j=n} p_{n,j}^{|S^m|_j} / |S^m|_j!$$

Furthermore, it is clear that the probability P_{m,n,S^m} is maximal if S^m is equal to the "normal" multiset Σ^m, which follows a normal distribution (*i.e.* $|\Sigma^m|_i = m \times p_{n,i}$). Thus, for any multiset S^m, $P_{m,n,S^m} \leq P_{m,n,\Sigma^m}$.

We can conclude that the number of solutions of an instance of $\mathcal{PPP}$, for a given multiset S^m, can be approximated by $N(m,n) \times P_{m,n,S^m}$ and therefore upper-bounded by $N(m,n) \times P_{m,n,\Sigma^m}$.

On figure 2, one can check the correctness of this evaluation by comparing its answers with some concrete tests:

- $N_{\mathcal{PP}}$ and P_{Σ^m} are the above evaluations for a given size $m \times n$;
- $E[N_{\mathcal{PPP}}]$ is an upper-bound on the expected number of solutions for a $\mathcal{PPP}$ instance of size $m \times n$ from the above analysis;
- $E_C[N_{\mathcal{PP}}]$ and $E_C[N_{\mathcal{PPP}}]$ are the average numbers of solutions over 50 tests on instances of size $m \times n$.

m	n	$N_{\mathcal{PP}}$	P_{Σ^m}	$E[N_{\mathcal{PPP}}]$	$E_C[N_{\mathcal{PP}}]$	$E_C[N_{\mathcal{PPP}}]$
11	17	148	0.017	3	181	3
13	17	65	0.013	1	76	1
13	19	212	0.010	2	238	2
15	19	65	0.007	1	82	1
17	21	107	0.004	1	141	1
19	23	105	0.003	1	119	1
21	25	160	0.002	1	170	1

Figure 2. Some Examples

If we want only one solution (or at least very few), on average, we have to choose m and n such that the product $N(m,n) \times P_{m,n,\Sigma^m}$ is close to 1. We have presented such values on figure 3. We can notice that the "good" sizes, around 100, are of the form $n = m + 16$.

4.2. Attacks against the Perceptrons Problem

4.2.1. A New Coding Let us now focus on how to solve the Perceptrons Problem. We can first remark that $\mathcal{PP}$ can be transformed into an equivalent problem more

suitable for implementations. It operates on 0-1 bits, instead of $+1$ and -1, and exclusive OR together with Hamming weight w_H, instead of dot product.

Problem 4
Given: *a binary matrix A of size $m \times n$,*
Question: *find a binary vector V such that $w_H(A_i \oplus V) \leq n/2$ for all i.*

This does not lead to an attack of the problem but it gives a way to improve the computations, using the native internal parallelism of the XOR operation, on 32-bit and 64-bit processors.

4.2.2. Approximation of the Solution Even if the theoretical analysis of MAX-$\mathcal{PP}$ proves that it is difficult to approximate (see Theorem 2), we can try to find a vector M such that $A \cdot M$ has many positive coordinates. The first one which comes to mind, and that we call the *majority vector*, consists of taking $M_j = +1$ if there are more components $A_{i,j}$ equal to $+1$ than to -1, and $M_j = -1$ otherwise. The following analysis proves two paradoxical results. First, the majority vector does not differ so much from the solution we are looking for. But, even if we know that a large part of the coordinates is correct, the number of vectors that still have to be tested in order to find a solution is very large, even if the parameters m and n are small.

LEMMA 1 *The Hamming distance between the solution of an instance and the majority vector is, on average, $n \times \left(\frac{1}{2} - \frac{1}{\pi}\sqrt{\frac{m}{n}}\right)$.*

Proof: We consider the same probability for A and V as described in section 4.1, namely: one randomly chooses an ε-matrix A' and an ε-vector V and sets $A_i = A'_i$ if $A_i \cdot V > 0$ and $A_i = -A'_i$ otherwise. If m and n are chosen in order to have, on average, one solution per $\mathcal{PPP}$-instance, we can say that V is "the" solution.

Let us first recall that we assume that $m = 2q + 1$ and $n = 2p + 1$. Let us compare V and the majority vector M. The probability for V and A'_i to have exactly r identical coordinates is $\frac{1}{2^n}\binom{n}{r}$. Consequently, the average number of identical coordinates between V and A_i is equal to $\frac{1}{2^{n+1}} \times \sum_{2r>n} r\binom{n}{r}$. Using the

m	n optimal	$N(m,n)$	P_{m,n,Σ^m}
101	117	9.4×10^{9}	8.3×10^{-11}
121	137	1.7×10^{11}	8.6×10^{-12}
151	169	2.1×10^{13}	5.9×10^{-14}
171	187	1.7×10^{14}	5.0×10^{-15}
191	207	2.3×10^{15}	4.1×10^{-16}
201	217	8.7×10^{15}	1.2×10^{-16}

Figure 3. Optimal Dimensions for $\mathcal{PPP}$

well known identities

$$\sum_{i=0}^{p} \binom{2p+1}{i} = \sum_{i=p+1}^{2p+1} \binom{2p+1}{i} = 2^{2p}$$

and

$$\sum_{i=p+1}^{2p+1} i \binom{2p+1}{i} = \frac{2p+1}{2}\left(2^{2p} + \binom{2p}{p}\right),$$

and the approximation

$$2^{-n}\binom{2p}{p} \approx \frac{1}{\sqrt{2\pi n}},$$

which comes from the Stirling's approximation of $n!$, we obtain that the probability F_n that $A_{i,j} = V_j$ is

$$F_n = \frac{1}{n} \times 2^{-(n+1)} \sum_{2r>n} r\binom{n}{r} = \frac{1}{2} + 2^{-n}\binom{2p}{p} \approx \frac{1}{2} + \frac{1}{\sqrt{2\pi n}}.$$

Let $G_{m,n}$ be the probability to have more than one half of the $\{A_{i,j}\}_{i<m}$ equal to V_j for any j. It can be computed, using again above equations and approximations in the following way:

$$\begin{aligned} G_{m,n} &= \sum_{s=q+1}^{2q+1} \binom{m}{s} (1-F_n)^{m-s} {F_n}^s \approx \frac{1}{2^m} \sum_{s=q+1}^{2q+1} \binom{2q+1}{s}\left(1 + \frac{2s-m}{\sqrt{\pi p}}\right) \\ &\approx \frac{1}{2^m}\left(1 - \frac{m}{\sqrt{\pi p}}\right) 2^{m-1} + \frac{1}{2^m}\frac{2}{\sqrt{\pi p}}\frac{m}{2}\left(2^{m-1} + \binom{2q}{q}\right), \end{aligned}$$

which can be approximated by $\frac{1}{2} + \frac{m}{2\pi\sqrt{pq}}$. In conclusion, the Hamming distance between the solution V and the majority vector M is about $n \times \left(\frac{1}{2} - \frac{1}{\pi}\sqrt{\frac{m}{n}}\right)$. ■

For example, with the sizes suggested on figure 3, this shows that about 80% of the coordinates of the majority vector are correct. So a first way to solve $\mathcal{PP}$ consists of computing the majority vector and then testing vectors which differ in less than 20% of the coordinates. But there are more than $\binom{n}{0.2\times n}$ such vectors and as soon as $n \geq 91$ the time complexity of this attack is greater than 2^{64}. We can try to improve this algorithm by first modifying the coordinates corresponding to columns which number of $+1$ is not very different from the number of -1. But many wrong coordinates of the majority vector come from non-ambiguous cases so that the efficiency is not really improved.

4.2.3. A Deterministic Attack Another idea to solve $\mathcal{PP}$ is based on trying to use the structure of the matrix A and to perform a kind of Gaussian reduction.

More precisely, if $A_i \cdot V > 0$ and $A_j \cdot V > 0$ then $(A_i + A_j) \cdot V > 0$. Furthermore, the vector $(A_i + A_j)/2$ has on average $n/2$ null coordinates. From all the pairs of rows of A, a system of $m(m-1)/2$ inequalities is obtained and after the permutation of the rows and columns we can hope to obtain a system which looks a bit like a triangular one (see figure 4, on the left part).

For more concreteness, let us focus on the example $(m, n) = (101, 117)$. We obtain a system of $m(m-1)/2 = 5050$ equations. The average number of rows with k zeros is $\frac{m(m-1)}{2} \frac{1}{2^n} \binom{n}{k}$. As a consequence, the average number of rows with $k = 77$ null coordinates is approximatively one. So, with high probability, there is a row with 77 null coordinates. It is chosen as the first inequality and the columns are permuted in order to have all the zeros ahead. Then, we can make the same analysis with the remaining rows, considering only the first 77 coordinates. This leads, with high probability, to a second row with 54 zeros ahead. Finally, we can expect to obtain a system of the form described in figure 5.

Once we have this system, we enumerate all the vectors of $n - 77 = 40$ coordinates obtained by modifying at most a fixed proportion p of the last 40 coordinates of the majority vector (see figure 4, on the right part). For all those vectors which product with the first row is positive (this happens with probability about $1/2$), we enumerate the $77 - 54 = 23$ previous coordinates in the same way and so on.

Finally, this leads to an algorithm that solves the problem with an expected number of elementary operations (product of two vectors) equal to

$$\binom{40}{p \times 40}\left[1 + \frac{1}{2}\binom{23}{p \times 23}\left[1 + \frac{1}{2}\binom{14}{p \times 14}[\ldots]\right]\right]$$

The value of the proportion p of coordinates that have to be changed have been statistically estimated and we obtained a complexity roughly equal to 2^{81}. Furthermore, the algorithm can solve one instance out of ten with complexity 2^{61}.

We investigated some ways to improve this deterministic algorithm. For example the fixed proportion of modified coordinates of the majority vector can vary from one step to another. Furthermore, one can try to add more than two rows of A to obtain rows with more than 77 zeros. However, we did not really improve the efficiency of the algorithm with such modifications.

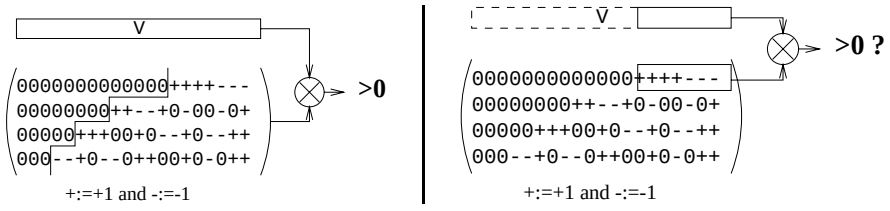

Figure 4. A Deterministic Algorithm to Solve $\mathcal{PP}$

row	1	2	3	4	5	6	7
number of zeros ahead	77	54	40	31	25	21	19

Figure 5. Expected System

4.2.4. A Probabilistic Attack Another way to solve $\mathcal{PP}$ is to use a probabilistic algorithm which tries to find a solution by successive approximations. It starts with a random vector and modifies it in order to reduce the Hamming distance to the solution. Many such algorithms are known, from the simple gradient descent to genetic algorithms [21, 8, 16]. We have chosen to implement the *simulated annealing* method but, because of the general lack of theoretical understanding of its mechanism, we can just give the actual results we obtained. However, we will see in section 4.3 that this is actually not a real problem.

We first need to define the neighborhood $\mathrm{Ngb}(V)$ of any ε-vector V, *i.e.* the vectors that are considered not to be very different, and an energy function $E(V)$ which quantifies the distance from the vector V to a solution of $\mathcal{PP}$. The algorithm starts with a random vector V and chooses a vector V' in its neighborhood. If the energy of V' is less than $E(V)$, V' becomes the current vector and so on. In order to avoid local minima, V' can also replace V even if its energy is higher but the probability of such an event decreases with time, like the temperature during metal annealing. A detailed description is presented on figure 6. It is parameterized by the initial temperature Θ_i, the final one Θ_f and the rate τ.

1. Choose a random vector V
2. Let $\Theta = \Theta_i$
3. Choose $V' \in \mathrm{Ngb}(V)$
4. Let $\Delta = E(V') - E(V)$
5. If $\Delta > 0$ then $p = exp(-\Delta/\Theta)$
   else $p = 1$
6. With probability p, let $V = V'$
7. Let $\Theta = \Theta \times \tau$
8. If $E(V) > 0$ and $\Theta > \Theta_f$ go to 3.

Figure 6. Simulated Annealing Algorithm

Many energy functions can be used. For example, the number of indices i such that $A_i \cdot V < 0$ is a possible choice, but the sum of $-A_i \cdot V$ over the indices such that $A_i \cdot V < 0$ (*i.e.* $-\sum_{i, A_i \cdot V < 0} A_i \cdot V$) is much more efficient. The neighborhood of any vector can be defined as the vectors obtained by changing only one coordinate.

As an example, with our implementation, the average time to find a solution for a 101×117-size instance is 3.5 seconds with an old 50 MHz Sparc 10.

Recently, Knudsen and Meier [23] improved this attack, finding better energy functions. As a consequence, they obtained an algorithm that solves $\mathcal{PP}$ 280 times

faster than was estimated in [32]. This shows how it is difficult to tune the parameters of such probabilistic algorithms.

4.3. Attacks against the Permuted Perceptrons Problem

At first sight, the Permuted Perceptrons Problem gives much more information about the solution than the basic Perceptrons Problem. In fact, it is straightforward to obtain one bit of the solution based on the public data but we have not been able to use more efficiently the knowledge of $\{\{S_i\}\}$. Let α (resp. α_i) be the number of coordinates of V (resp. A_i) equal to $+1$. We have $A_i \cdot V = n + 2\alpha + 2\alpha_i \bmod 4$. Consequently, from $S_i \bmod 4$, we learn $\alpha \bmod 2$, *i.e.* one bit of the private key.

$\mathcal{PPP}$ is hard to solve because of the permutation of the values S_i. If this permutation were known, a single Gaussian reduction of the matrix would lead to the solution but the guess of this permutation seems computationally infeasible. Furthermore, if we want to use simulated annealing algorithms or genetic algorithms to directly solve $\mathcal{PPP}$, we have to find a good energy function (or fitness function). One which should suit well is defined as follows: $E(V) = \sum_i |T_i - R_i|$, where $T = sort(S)$ and $R = sort(A \cdot V)$. But, on average, the energy difference between two neighbors is similar to the energy difference between two random points. It therefore does not provide any kind of continuity, which does not help the algorithm to converge to the solution. So, the only way to solve $\mathcal{PPP}$ seems to get a solution of the related Perceptrons Problem and test whether it solves $\mathcal{PPP}$ or not. It may be possible to make the Perceptrons Problem algorithm to more likely find a $\mathcal{PPP}$ solution than any one else, as attempted to do Knudsen and Meier [23], but the improvement is not so significant.

For suggested parameters (m, n), an instance of $\mathcal{PPP}$ has about one solution but there are so many solutions for the related Perceptrons Problem that, even if we are able to compute them efficiently, the time needed to find the good one is still very large.

According to our experiments and those of Knudsen and Meier, we therefore advise $(m, n) = (121, 137)$ as a minimal choice, and may be $(201, 217)$ for very secure applications.

Note. When the problem $\mathcal{PP}$ is modified into $\mathcal{PPP}$, it could seem surprising to keep the positivity constraint on the values S_i. The basic reason is that this enables to have instances with, on average, just one solution. After such an observation, and since known attacks against $\mathcal{PPP}$ are derived from attacks on $\mathcal{PP}$, we could imagine to choose another constraint on the S_i to reduce the average number of solutions, for example $|S_i| > k$ instead of $S_i > 0$. But, doing this, we would no longer be able to generate instances with known solutions.

5. Application to Cryptography

Since small sizes of the parameters are enough to get a difficult problem, we propose to use it for identification protocols. So, we present two schemes which security is equivalent to the Permuted Perceptrons Problem.

5.1. The Three-Pass Identification Protocol

We first present a three-pass protocol, for a given security parameter k. Let us describe this scheme in two phases: the initialization and the identification. In $\mathcal{PPP}$, the arithmetical operations are performed in $\mathbb{Z}$, however, the protocol needs to perform those operations in a finite set, the modular finite ring $\mathbb{Z}_p$. Therefore, we have to determine the smallest value of p such that the reduction modulo p of the operations does not modify the problem. More precisely, let V be an ε-vector of odd size n and S be a positive and odd integer. We are looking for p such that $V \cdot Y = S$ is equivalent to $V \cdot Y = S \bmod p$, for any ε-vector Y. Since $V \cdot Y$ is odd, we require that $V \cdot Y = S + 2kp \Longleftrightarrow V \cdot Y = S$. Furthermore, $|V \cdot Y - S| < 2n$ so the equivalence occurs as soon as $p > n$.

Then, we have a public integer $p > n$ which defines the ring in which all the computations will be done, together with a public hash function H. Next, any user chooses his public and private keys, (A, S) and V as described above.

The global identification scheme consists of k sequential iterations of the following protocol, where any random choice is performed uniformly in the finite space:

- the Prover chooses a random permutation P to permute the rows of A, and a random signed permutation Q, to permute and possibly change the sign of the columns. Then, he chooses a random vector W in $(\mathbb{Z}_p)^n$.
- the Prover computes the new instance $A' = PAQ$, with the new solution $V' = Q^{-1}V$. The multi-set remains unchanged.
- the Prover masks V' with the (previously randomly chosen) vector W into the vector $R = V' + W$.
- Then, he begins interactions with the Verifier:
  - the Prover computes the commitments of all the elements randomly chosen above $h_0 = H(P|Q)$ and $h_1 = H(W)$ as well as the following commitments, $h_2 = H(R)$, $h_3 = H(A'W)$ and $h_4 = H(A'R)$. Then, he sends all of them, h_0, h_1, h_2, h_3 and h_4, to the Verifier.
  - the Verifier chooses a random question $c \in \{0, 1, 2, 3\}$, and sends it to the Prover.
  - the Prover answers according to the query c

    if $c = 0$, he sends P, Q and W;

    if $c = 1$, he sends P, Q and R;

if $c = 2$, he sends $A'W$ and $A'V'$;
if $c = 3$, he sends the vectors W and V';

– the Verifier accepts after having checked, according to his query c:
for $c = 0$, if $h_0 = H(P|Q)$, $h_1 = H(W)$ and $h_3 = H(PAQW)$;
for $c = 1$, if $h_0 = H(P|Q)$, $h_2 = H(R)$ and $h_4 = H(PAQR)$;
for $c = 2$, if $h_3 = H(A'W)$, $h_4 = H(A'W + A'V')$ and $\{\{(A'V')_i\}\} = S$;
for $c = 3$, if $h_1 = H(W)$, $h_2 = H(W + V')$ and $V' \in \{-1, +1\}^n$.

If the Verifier accepts the k iterations, then he accepts the identification, otherwise, he rejects it.

We first state a lemma which proves the soundness of this interactive proof system.

LEMMA 2 *Let $\mathcal{A}$ be a probabilistic polynomial time Turing machine which can perform an impersonation with probability $\pi = \left(\frac{3}{4}\right)^k + \nu$, for some $\nu > 0$. Then there is another machine which has control over $\mathcal{A}$ and solves the Permuted Perceptrons Problem, or finds a collision for the hash function, with probability greater than $3\nu^2/14$ after less than $1 + 4k$ calls to $\mathcal{A}$.*

Proof: This proof is mostly inspired by [46], where one assumes that an attacker $\mathcal{A}$ can perform an impersonation with probability $\pi = \left(\frac{3}{4}\right)^k + \nu$ for some $\nu > 0$. Let us denote by ω and ω' the random tapes of this adversary and of the Verifier respectively, and by I the list of the challenges asked by the Verifier. It is clear that I is a random variable which may depend on both ω and ω'. However, in that proof, we only consider honest verifiers who uniformly, and independently from anything else, chooses I. We can therefore identify ω' and I. Let us denote by $\mathcal{S}$ the set of the pairs (ω, I) which lead to acceptance: $\Pr[(\omega, I) \in \mathcal{S}] = \pi$. Let us define the set $\Omega = \{\omega \,|\, \Pr_I[(\omega, I) \in \mathcal{S}] \geq \pi - \nu/2\}$. If we assume that $\Pr[\Omega] < \nu/2$, then

$$\pi = \Pr[\mathcal{S}] = \Pr[\mathcal{S}\,|\,\Omega] \cdot \Pr[\Omega] + \Pr[\mathcal{S}\,|\,\bar{\Omega}] \cdot \Pr[\bar{\Omega}] < \nu/2 + (\pi - \nu/2) = \pi,$$

which implies a contradiction, so $\Pr[\Omega] \geq \nu/2$. Furthermore, using Bayes' law one can show that

$$\Pr[\Omega\,|\,\mathcal{S}] = 1 - \Pr[\bar{\Omega}\,|\,\mathcal{S}] = 1 - \Pr[\mathcal{S}\,|\,\bar{\Omega}] \times \Pr[\bar{\Omega}]/\Pr[\mathcal{S}] \geq 1 - (\pi - \nu/2)/\pi = \nu/2\pi.$$

Consequently, if we run an attack for randomly chosen (ω, I), with probability greater than π, $(\omega, I) \in \mathcal{S}$. In that latter case, with probability $\nu/2\pi$, we furthermore have $\omega \in \Omega$. Let us assume that holds in the following of the proof, therefore, $\Pr_I[(\omega, I) \in \mathcal{S}] \geq \pi - \nu/2$. Let us consider the execution tree $T(\omega)$ corresponding to all accepted I, with above ω. Using arguments analogous to [46], we denote by n_i the number of nodes at depth i. We know that $n_0 = 1$ and $n_k = 3^k + 4^k\nu/2$ (since $n_k/4^k = \Pr_I[(\omega, I) \in \mathcal{S}] \geq \pi - \nu/2$). Using a convexity relation on the logarithm of the following relation,

$$\prod_{i=0}^{k-1} \frac{n_{i+1}}{n_i} = \frac{n_k}{n_0} \geq 3^k + \frac{\nu}{2} \cdot 4^k \geq (1 - \frac{\nu}{2}) \cdot 3^k + \frac{\nu}{2} \cdot 4^k,$$

one obtains

$$\sum_{i=0}^{k-1} \log \frac{n_{i+1}}{n_i} \geq (1-\frac{\nu}{2}) \log 3^k + \frac{\nu}{2} \log 4^k \geq k\Big(\log 3 + \frac{\nu}{2} \log(4/3)\Big).$$

Hence, there exists $i < k$ such that

$$\frac{n_{i+1}}{n_i} \geq 3 \cdot (4/3)^{\nu/2} = 3 \cdot e^{\frac{\nu}{2} \cdot \log(4/3)} \geq 3 \cdot \Big(1 + \frac{\nu}{2} \cdot \log(4/3)\Big) \geq 3 \cdot \Big(1 + \frac{\nu}{7}\Big).$$

Let us respectively denote by f_i and t_i the number of nodes at depth i with exactly 4 sons and the number of nodes at depth i with at most 3 sons:

$$n_i = f_i + t_i \text{ and } n_{i+1} \leq 4f_i + 3t_i = f_i + 3n_i.$$

Therefore, $3 + f_i/n_i \geq 3 + 3\nu/7$, which implies $f_i/n_i \geq 3\nu/7$. And so, with probability greater than $3\nu/7$, the path I contains a node with 4 sons. In that case, we can find it by trying the $4k$ possible nodes along this path.

Finally, after less than $4k + 1$ calls to the machine $\mathcal{A}$, with probability greater than $\pi \times \nu/2\pi \times 3\nu/7 = 3\nu^2/14$, we have found a node with 4 sons. Such a node corresponds to the situation where the five commitments h_0, h_1, h_2, h_3 and h_4 have been made and the attacker can answer the four questions of the verifier:

$$\begin{array}{lrcccll}
c=0 & H(P_0|Q_0) & = & h_0 & = & H(P_1|Q_1) & c=1 \quad (1)\\
c=0 & H(W_0) & = & h_1 & = & H(W_3) & c=3 \quad (2)\\
c=1 & H(R_1) & = & h_2 & = & H(W_3 + V_3') & c=3 \quad (3)\\
c=0 & H(P_0AQ_0W_0) & = & h_3 & = & H(Y_2) & c=2 \quad (4)\\
c=1 & H(P_1AQ_1R_1) & = & h_4 & = & H(Y_2 + Z_2) & c=2 \quad (5)
\end{array}$$

Unless we have found a collision for the hash function H, we can consider that

$$\begin{array}{lcl}
(1) \Rightarrow & P = P_0 = P_1, & \\
(1) \Rightarrow & Q = Q_0 = Q_1, & \\
(2) \Rightarrow & W = W_0 = W_3, & \\
(3) \Rightarrow & R = R_1 = W_3 + V_3' = W + V' \bmod p, & \\
(4) \Rightarrow & Y = Y_2 = P_0AQ_0W_0 = PAQW \bmod p, & \\
(5) \Rightarrow & Y + Z = Y_2 + Z_2 = P_1AQ_1R_1 = PAQR \bmod p &
\end{array}$$

such that $V' \in \{-1, +1\}^n$ and $\{\{Z_i\}\} = S \bmod p$. Then,

$$Y + Z = PAQR = PAQW + Z = PAQW + PAQV' \bmod p,$$

consequently, $Z = PAQV' \bmod p$. If we let $V = QV'$, we have $Z = PAV \bmod p$, hence $\{\{(AV)_i\}\} = S \bmod p$. Since $p > n$, we have also $\{\{(AV)_i\}\} = S$. Finally, we have solved the Permuted Perceptrons Problem instance in time $4k + 1$, with probability greater than $3\nu^2/14$. ■

Hence the following Theorem:

THEOREM 4 *If $p > n$, this protocol is an interactive proof system for $\mathcal{PPP}$.*

Proof: This protocol is clearly complete, and the previous lemma proves the soundness: if there exist a polynomial P and an attacker $\mathcal{A}$ which can perform an impersonation, for any security parameter k, in time t, with probability greater than $(3/4)^k + 1/P(k)$, then there exists a machine which can either extract the secret key or find a collision, in time $(4k+1) \times t$, with probability greater than $3/14P^2(k)$. ■

We now assume that $m \leq n$ and that the rank of the matrix A is equal to m, *i.e.* there exists an $m \times m$-sub-matrix K of A with an invertible determinant in $\mathbb{Z}_p$.

THEOREM 5 *In the random oracle model, or with a secure commitment scheme, if the rank of the matrix A is equal to m, this interactive proof system is zero-knowledge.*

Proof: We want to prove that the interaction between the prover and a possibly crooked verifier can be simulated by a probabilistic polynomial time Turing machine without the secret, with an indistinguishable distribution.

Let us denote by Σ the strategy of the crooked verifier: for any history tape $hist$, and the view of the five commitments h_0, h_1, h_2, h_3, h_4, this strategy is used to get an optimal query c: $\Sigma(\omega, hist, h_0, h_1, h_2, h_3, h_4)$ outputs a query between 0 and 3. In the case of an honest verifier, this strategy would be independent of $hist, h_0, h_1, h_2, h_3, h_4$, but more generally we consider a crooked one.

Let us describe a probabilistic polynomial time Turing Machine $\mathcal{S}$ which builds communication tapes with an indistinguishable distribution from the real ones. Once again, any random choices are performed according to uniform distributions in the respective finite sets.

1. $\mathcal{S}$ chooses a uniformly random query $C \in \{0, 1, 2, 3\}$

   if $C = 0$, $\mathcal{S}$ randomly chooses P, Q and W. It computes the commitments $h_0 = H(P|Q)$, $h_1 = H(W)$ and $h_3 = H(PAQW)$ and randomly chooses the strings h_2 and h_4. Then, it lets $x = (h_0, h_1, h_2, h_3, h_4)$ and $y = (P, Q, W)$.

   if $C = 1$, $\mathcal{S}$ randomly chooses P, Q and R. It computes the commitments $h_0 = H(P|Q)$, $h_2 = H(R)$ and $h_4 = H(PAQR)$ and randomly chooses the strings h_1 and h_3. Then, it lets $x = (h_0, h_1, h_2, h_3, h_4)$ and $y = (P, Q, R)$.

   if $C = 2$, $\mathcal{S}$ randomly chooses a vector Y such that $\{\{Y_i\}\} = S$, and another vector X. Y is supposed to be PAV and X is supposed to be $PAQW$. It is clear that Y follows the same distribution as PAV. But what about the distribution of $PAQW$? Let $Z \in (\mathbb{Z}_p)^m$. We assume that $Y = PAV$ is fixed, which fixes the permutation P. The probability for $PAQW$ to be equal to Z, over uniformly distributed Q and W, is equal to

$$\frac{\sum_Q \#\{W | AQW = P^{-1}Z\}}{2^n n! p^n} = \frac{\sum_Q \#\{W | AW = P^{-1}Z\}}{2^n n! p^n},$$

since the Q are invertible. Randomly choosing the $(n-m)$ coordinates, not corresponding to the sub-matrix K, there is only one solution for the m others. Then the previous probability is equal to $\sum_Q p^{n-m}/(2^n n! p^n) = 1/p^m$. $\mathcal{S}$ computes the commitments $h_3 = H(X)$, $h_4 = H(X+Y)$ and randomly chooses the strings h_0, h_1 and h_2. Then, it sets $x = (h_0, h_1, h_2, h_3, h_4)$ and $y = (X, X+Y)$.

if $C = 3$, $\mathcal{S}$ randomly chooses a vector W and an ε-vector E, computes the commitments $h_1 = H(W)$, $h_2 = H(W+E)$ and randomly chooses the strings h_0, h_3 and h_4. Then, it lets $x = (h_0, h_1, h_2, h_3, h_4)$ and $y = (W, W+E)$.

2. $\mathcal{S}$ evaluates $c = \Sigma(\omega, hist, x)$.

3. If $c = C$ then $\mathcal{S}$ writes x, c and y, otherwise $\mathcal{S}$ rewinds the history tape and goes back to 1 (reset [18]).

Consequently, if H is a random oracle [3], or represents a secure commitment scheme [7, 20], $\mathcal{S}$ simulates communication tapes with an indistinguishable distribution from a real identification of k rounds after an expected number of steps bounded by $4 \times k$. ■

These results prove the security of this identification scheme even against active attacks relative to the Permuted Perceptrons Problem.

5.2. The Five-Pass Identification Protocol

As we have seen, the probability of impersonation is only upper bounded by 3/4 at each round. In order to obtain an acceptable security level (say probability of impersonation less than one over a million), we have to iterate the protocol 48 times.

The following protocol, the five-pass one, provides a better security level, since the probability of impersonation is roughly upper bounded by 2/3 at each round.

For any security parameter k, the initialization is the same as before. The identification consists of k repetitions of the following protocol, where any random choices follow uniform distributions:

- the Prover chooses a random permutation P to permute the rows of A and a random signed permutation Q to permute and possibly change the sign of the columns. Then he chooses a random vector W in $(\mathbb{Z}_p)^n$.

- the Prover computes the new instance $A' = PAQ$, together with the new solution $V' = Q^{-1}V$. The multi-set remains unchanged.
  Until this stage, everything is the same as in the three-pass protocol, but the Prover waits before computing R.

- Then, he begins interactions with the Verifier:

  – the Prover computes the commitments $h_0 = H(P|Q)$, $h_1 = H(W|V')$ and $h_2 = H(A'W|A'V')$, and sends them to the Verifier.

- the Verifier chooses a random t between 1 and $p-1$, and sends it to the Prover.
- the Prover masks V' with W into the vector $R = tW + V'$ and computes $h_3 = H(R)$ and $h_4 = H(A'R)$. He sends h_3 and h_4 to the Verifier.
- the Verifier chooses a random question $c \in \{0, 1, 2\}$, and sends it to the prover.
- the Prover answers according to the query c

  if $c = 0$, he sends P, Q and R;
  if $c = 1$, he sends $A'W$ and $A'V'$;
  if $c = 2$, he sends W and V'.
- the Verifier accepted after having checked, according to his query c

  for $c = 0$, if $h_0 = H(P|Q)$, $h_3 = H(R)$ and $h_4 = H(PAQR)$;
  for $c = 1$, if $h_2 = H(A'W|A'V')$, $h_4 = H(tA'W + A'V')$ and the multi-set $\{\{(A'V')_i\}\}$ is equal to S;
  for $c = 2$, if $h_1 = H(W|V')$, $h_3 = H(tW + V')$ and $V' \in \{-1, +1\}^n$.

If the Verifier accepts k iterations, then he accepts the identification, otherwise, he rejects it.

THEOREM 6 *If p is a prime number greater than n, this protocol is an interactive proof system for $\mathcal{PPP}$.*

Proof: The same proof as for the three-pass scheme establishes this Theorem. However, for the soundness, we need p to be prime, so that any non-zero element in $\mathbb{Z}_p$ is invertible.

In the reduction, we try to extract two distinct t for which the adversary can answer the three values for the challenge c. This is not possible if the adversary can answer the three values of c for just one t, and just two challenges for the other t: which represents $3 + 2(p-2)$ successful situations, and therefore $2(p-1)+1$ among the $3(p-1)$ possible challenges. One more possible success would give us the possibility to extract what we are looking for.

LEMMA 3 *If there exist a polynomial P and an attacker A which can perform an impersonation, for any security parameter k (the number of iterations), with probability greater than $((2p-1)/(3(p-1)))^k + 1/P(k)$, then there exists a machine which can either extract the secret key or find a collision, after less than $3pk$ calls to A, with probability greater than $3/14P^2(k)$.*

That Lemma proves the Theorem. ■

THEOREM 7 *In the random oracle model, or with a secure commitment scheme, if the rank of the matrix A is equal to m, for any fixed prime p greater than n, this interactive proof system is zero-knowledge.*

6. Practical Identification Scheme

In this section, we first study secure and efficient strategies to choose the system parameters and the "per-entity" secrets. Next we describe a practical version of the identification schemes theoretically analyzed in the previous section. Then, we summarize the results we obtained with an actual implementation of this protocol on a low-cost smart card. Finally, we compare the Permuted Perceptrons based schemes with others also based on $\mathcal{NP}$-complete problems.

6.1. Selection of the System Parameters

We first need to choose the parameters m and n. In order to have on average one solution per instance, those parameters have to verify the heuristic relation $n \approx m + 16$. The choice of one of the convenient pairs (m, n) will immediately fix the level of security of the underlying problem. As a conclusion of the analysis of section 4.3, we can advise $(m, n) = (121, 137)$ as a minimal choice and larger sizes, as $(m, n) = (201, 217)$ for a higher level of security (since it seems a million times harder to solve).

Next, as we have previously seen, we need to perform mathematical operations in a finite field in order to hide information. So, we have to choose a prime $p > n$.

Finally, let A be a publicly known ε-matrix. The public keys of all the users will be derived from this single matrix. For security and efficiency reasons, the matrix can be generated in the following way: let g_0 be a seed used by a pseudo-random generator to produce m secondary seeds $g_1, \ldots, g_m$. Then each g_i is used to produce the ith row A_i of the matrix A using again a pseudo-random generator. This way, the matrix can be stored using about 10 bytes and furthermore this is a guarantee of its (pseudo)-randomness.

6.2. Selection of "Per- Entity" Secrets

Each user can choose his own private key V and compute the related public key. To achieve this goal, he computes the unique vector L, defined by $L_i = sign(A_i \cdot V)$ for all i, and the multi-set $S = \{\{L_i \times (A_i \cdot V)\}\}$.

We now show how to store efficiently the public key (L, S). First notice that, even if the elements of S can be as large as n, usually, they are much smaller so we can assume that they are less than a constant t. Furthermore, the average number of elements of S equal to k is $\frac{m}{2^{n-1}} \binom{n}{(n+k)/2}$. Let us define the function

$$Average(m, n, k) = \left\lceil \frac{m}{2^{n-1}} \binom{n}{(n+k)/2} \right\rfloor,$$

where for any real number x, $\lceil x \rfloor$ denotes the nearest integer from x. The values of this function for $(m, n) = (121, 137)$ are shown on figure 7. We can assume that for any odd k less than t, the number of elements of S equal to k lies in $[Average(m, n, k) - 2^{\ell-1}, Average(m, n, k) + 2^{\ell-1}[$. Figure 8 presents the fraction

k	1	3	5	7	9	11	13	15
Average	16	16	15	14	12	11	9	7

k	17	19	21	23	25	27	29	≥ 31
Average	6	4	3	2	2	1	1	0

Figure 7. Function *Average* for $(m, n) = (121, 137)$

(in percents) of the instances which can be coded for given t and ℓ, for the parameters $(m, n) = (121, 137)$. Then, the second table shows that, if we just consider an half of the keys, the public key can be stored very efficiently.

Of course this technique does not enable to use any instance as a pair of secret and public keys but one can remark that the excluded instances are very special cases, which are probably easier to break. For example, if too many products $A_i \cdot V$ have identical values, this drastically reduces the difficulty of finding a permutation of the set S equal to $A \cdot V$. Furthermore, if an element of S is too large, it means that V is not very different from one of the $L_i A_i$'s, so it may be found quite easily. Therefore, the optimization of key storage is not a drawback but it is even a way to avoid the choice of weaker keys.

6.3. Practical Protocol

We now describe a practical version of the three-pass identification scheme. It differs from the original one in the use of hash trees as commitment and of pseudo-random generators. These techniques allow a huge reduction of the communication. We have chosen to focus on the three-pass version. Indeed, even if the communication load is a little larger than for the five-pass version, it is more efficient from a

Percentage of instances which can be coded for $(m, n) = (121, 137)$				
$t \backslash \ell$	3	4	5	∞
31	2	24	29	29
33	4	39	48	48
35	5	**54**	65	66
37	7	65	78	78
39	8	73	87	88
41	9	78	93	93
∞	9	84	99	100

m	n	t	ℓ	Key size (bits)	
				Public $(= \frac{\ell(t-1)}{2} + m)$	Secret $(= n)$
121	137	35	4	189	137
141	157	39	4	217	157
161	177	43	4	245	177
181	197	45	4	269	197
201	217	45	5	311	217

Figure 8. Practical Parameters

computational point of view and furthermore easier to implement: the size of the program is much smaller, which is important for an implementation on a smart-card.

Let us assume that Alice wants to identify herself to Bob. First she randomly chooses two seeds g_1 and g_2. The seed g_1 is used to generate the permutation P of $\{0,\ldots,m-1\}$ and the signed permutation Q of $\{0,\ldots,n-1\}$. The seed g_2 generates the vector W in $(\mathbb{Z}_p)^n$. Then she computes the following hash values: $h_0 = H(g_1)$, $h_1 = H(g_2)$, $h_2 = H(Q^{-1}V + W)$, $h_3 = H(PAQW)$ and $h_4 = H(PAQW + PAV)$. Finally she sends the single commitment, which can be seen as the root of a hash tree, $C = H(H(h_0, h_1, h_2), H(h_3, h_4))$ (see figure 9). Then, Bob answers a challenge

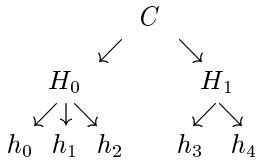

Figure 9. Hash Tree

$c \in \{0,1,2,3\}$, and Alice answers according to this query

- if $c = 0$, Alice sends g_1, g_2, h_2 and h_4.
- if $c = 1$, Alice sends g_1, $R = W + Q^{-1}V$, h_1 and h_3.
- if $c = 2$, Alice sends $X = PAQW$, $Y = PAV$ and $h_{012} = H(h_0, h_1, h_2)$.
- if $c = 3$, Alice sends g_2, $V' = Q^{-1}V$, h_0 and $h_{34} = H(h_3, h_4)$.

And Bob accepts if

- for $c = 0$, $C = H(H(H(g_1), H(g_2), h_2), H(H(A'W), h_4))$, where $A' = PAQ$.
- for $c = 1$, $C = H(H(H(g_1), h_1, H(R)), H(h_3, H(A'R)))$, where $A' = PAQ$.
- for $c = 2$, $C = H(h_{012}, H(H(X), H(X + Y)))$ and $S = \{\{Y_i\}\}$.
- for $c = 3$, $C = H(H(h_0, H(W), H(W + V')), h_{34})$ and $V' \in \{-1, +1\}^n$.

Let us denote respectively by $|x|$, $|H|$ and $|g|$ the number of bits needed to represent any x, the size of the hash values and the size of the seeds. Then, the number of bits transmitted during each round is $|H|$ for the first commitment, plus 2 bits for the query and

- if $c = 0$, $2|g| + 2|H|$.
- if $c = 1$, $|g| + 2|H| + n|p|$.

- if $c = 2$, $m|p| + m|(t-1)/2| + |H|$.
- if $c = 3$, $|g| + n + 2|H|$.

Globally, this amounts to $|H| + 2$ plus a quarter, on average, of the sum of each case: $4|g| + 7|H| + n(|p| + 1) + m(|p| + |(t-1)/2|)$, that is

$$\frac{11}{4}|H| + |g| + \frac{|p|+1}{4}n + \frac{|p| + |(t-1)/2|}{4}m + 2.$$

For $m = 121$, $n = 137$, $|p| = 7$, $t = 35$, $|H| = 128$ and $g = 120$, this is equal to 1084 bits. If the protocol needs 48 rounds to achieve a requested level of security, the average size of the communications with those parameters is 6.34 KBytes.

6.4. Pseudo-Random Generators

The identification protocol needs the random generation of mathematical objects. First, in order to randomly choose objects like vectors or permutations and then to store them efficiently, a very convenient technique consists of choosing a random seed and then generating an object of the accurate format with a pseudo-random generator. As a first application, we have already said that the matrix A was generated from a single seed. We just want the matrix to look random so we can use very simple pseudo-random generators like the truncated congruential linear one [24].

The same technique can also be used with the seeds g_1 and g_2 of the practical identification scheme to generate the objects needed to hide information. We insist on the fact that we do not need the pseudo-random generator to be cryptographically secure; we just need a procedure to pick mathematical objects "uniformly" in a large set. For example, if we use seeds of 64 bits to generate a permutation of $\{0, \ldots, 2^{117} - 1\}$, we immediately see that we cannot produce all the permutations but we want the subset of permutations that can be generated to look like any subset of cardinality 2^{64} randomly chosen in the large set. A very efficient way to generate pseudo-random permutations, studied by Luby and Rackoff [25], uses the well-known structure proposed by Feistel.

6.5. Actual Implementation on a Low-Cost Smart Card

We have implemented this protocol on a low-cost smart card based on a Motorola 6805 microprocessor running at 3.57 MHz. This is a very simple chip, performing classical operations on 8 bits. A smart card possesses three kinds of memory, a ROM of 4 KBytes, an EEPROM of 2 KBytes where the program and the static data are stored and a 160-Bytes RAM, among which only 120-Bytes can be used to store variables. The communication rate is 19200 bit/s and the verifier is a simple PC.

At first sight, due to the size of the objects, it seems difficult to implement the protocol with such a small RAM. Furthermore, the stack is automatically set in

the middle part of the RAM, and consequently the available memory is even not contiguous. But, using pseudo-random generators, it becomes possible to implement the scheme with sufficiently large parameters. We have chosen the parameters $m = 121$, $n = 137$ with $p = 127$, $t = 35$. Moreover, we have implemented a version of SNEFRU [38] as a hash function; it works with 12 bytes long blocks of data and is based on 4 repetitions of an elementary round. Even if this primitive cannot be advised for a secure application, the use of SHA-1 [26] or MD5 [39] in real applications would even save some EEPROM since they are implemented in ROM of most of the actual smart-cards.

Let us now focus on the performance. First notice that, whatever the performance may be, it is already an interesting result to prove the feasibility of implementing a zero-knowledge identification scheme without precomputation and just using an 8-bit microprocessor. It would be impossible to do such a thing with a number theoretical based protocol that needs to perform modular arithmetic with large integers.

In order to perform 48 rounds of identification and consequently to be able to identify a cheater with probability $1 - 10^{-6}$, the card needs 23 seconds of computation and the communication takes 5 seconds. So finally a complete identification needs less than 30 seconds. A few comments are in order. First, 40% of the computation time is used by the hash function and 35% by the computation of products of matrices and vectors. Furthermore, the implementation as been done with a low-end smart-card, presently obsolete, and an implementation on current low-cost smart cards would be much more efficient, mainly because the time needed by the computation could be greatly reduced

- because of a higher internal frequency (up to 10MHz)
- because of a faster communication rate (up to 115200 bit/s)
- with just a little more EEPROM.

Consequently, a complete authentication could be performed in about 10 seconds.

Finally, in many applications like pay-TV or pay-phone for which it does not matter if a lucky cheater is able to use a resource during a few seconds, the identification can be performed as a background job. In fact, each iteration of the protocol takes less than a second, and the RAM can be completely cleared between two iterations, and freed to another application.

7. Comparison with other Protocols

It is interesting to compare the known protocols based on non-number-theoretical, $\mathcal{NP}$-complete problems. The first one, PKP, as been proposed by Shamir [42]. Then Stern proposed SD [44, 46], a scheme based on error-correcting codes, and CLE [45]. The figure 10 summarizes the size of the public and secret keys and of the communication for those schemes with parameters chosen in such a way that the security is about the same [11, 1, 6, 30, 34], using hash functions of 128 bits like MD5 and seeds of 120 bits. The communication needed by PPP seems to be its

scheme	PKP		CLE		SD	PPP	
	3p	5p	3p	5p		3p	5p
matrix size	16×34		24×24		256×512	121×137	
on the field	$\mathbb{F}_{251}$		$\mathbb{F}_{257}$		$\mathbb{F}_2$	$\mathbb{F}_2$	
number of rounds	35	20	35	20	35	48	35
Public key size (bits)	272		96		256	189	
Secret key size (bits)	141		96		512	137	
Communication (KB)	2.20	1.63	2.41	1.95	3.62	6.34	6.40

Figure 10. Comparison of $\mathcal{NP}$-complete Problems based Identification Schemes in terms of communication complexity

main drawback but this is not very important for many applications. Furthermore, 6.4 KBytes take only 2.8 seconds to be transmitted at 19200 bit/s, and less than 0.5 second if transmitted at 115200 bit/s.

All those $\mathcal{NP}$-complete problem based schemes can be implemented on the smart card we used, i.e. using about 100 bytes of RAM and a few KB of EEPROM. In comparison, it would be much more difficult to implement classical schemes like Fiat-Shamir [9] or Schnorr [41] since they need the implementation of a modular exponentiation. Notice that some variants like GPS [13, 35] partially solve this problem by using precomputed coupons.

8. Conclusion

In this paper, we have presented a new combinatorial problem, the Permuted Perceptrons Problem. We have studied its complexity properties, both theoretically and practically. The results we have obtained prove that this problem is well suited for cryptographic applications. In fact, no efficient algorithm can solve it, even for small sizes. Furthermore, we have proposed two efficient zero-knowledge identification schemes which security relies on this problem. The performances of those protocols are comparable with those of other identification schemes based on $\mathcal{NP}$-complete problems.

Acknowledgments

We thank Louis Granboulan for the results about the majority vector, as well as Jacques Stern for having proposed the perceptrons problem as a candidate for efficient identification schemes and for many fruitful discussions. We also thank Russell Impagliazzo for discussions about probabilistic algorithms and the reviewers for their valuable feedback.

References

1. T. Baritaud, M. Campana, P. Chauvaud, and H. Gilbert. On the Security of the Permuted Kernel Identification Scheme. In *Crypto '92*, LNCS 740, pages 305–311. Springer-Verlag, Berlin, 1992.
2. E. B. Baum, D. Boneh, and C. Garrett. On Genetic Algorithms. In *Proc. of the 8th COLT*, pages 230–239. ACM Press, New York, 1995.
3. M. Bellare and P. Rogaway. Random Oracles Are Practical: a Paradigm for Designing Efficient Protocols. In *Proc. of the 1st CCS*, pages 62–73. ACM Press, New York, 1993.
4. E. F. Brickell and K. S. McCurley. An Interactive Identification Scheme Based on Discrete Logarithms and Factoring. In *Eurocrypt '90*, LNCS 473, pages 63–71. Springer-Verlag, Berlin, 1991.
5. E. F. Brickell and K. S. McCurley. An Interactive Identification Scheme Based on Discrete Logarithms and Factoring. *Journal of Cryptology*, 5:29–39, 1992.
6. F. Chabaud. On the Security of Some Cryptosystems Based on Error-Correcting Codes. In *Eurocrypt '94*, LNCS 950, pages 131–139. Springer-Verlag, Berlin, 1995.
7. I. B. Damgård, T. P. Pedersen, and B. Pfitzmann. On the Existence of Statistically Hiding Bit-Commitment Schemes and Fail-Stop Signatures. In *Crypto '93*, LNCS 773, pages 250–267. Springer-Verlag, Berlin, 1994.
8. L. Davis, editor. *Genetics Algorithms and Simulated Annealing*. Pitman, London, 1987.
9. A. Fiat and A. Shamir. How to Prove Yourself: Practical Solutions of Identification and Signature Problems. In *Crypto '86*, LNCS 263, pages 186–194. Springer-Verlag, Berlin, 1987.
10. M. R. Garey and D. S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness*. Freeman, San Francisco, CA, 1979.
11. J. Georgiades. Some Remarks on the Security of the Identification Scheme Based on Permuted Kernels. *Journal of Cryptology*, 5(2):133–137, 1992.
12. M. Girault. An Identity-Based Identification Scheme Based on Discrete Logarithms Modulo a Composite Number. In *Eurocrypt '90*, LNCS 473, pages 481–486. Springer-Verlag, Berlin, 1991.
13. M. Girault. Self-Certified Public Keys. In *Eurocrypt '91*, LNCS 547, pages 490–497. Springer-Verlag, Berlin, 1992.
14. M. Girault and J.-C. Paillès. An Identity-Based Identification Scheme Providing Zero-Knowledge Authentication and Authenticated Key Exchange. In *ESORICS '90*, LNCS, pages 173–184. Springer-Verlag, Berlin, 1990.
15. M. Girault and J. Stern. On the Length of Cryptographic Hash-Values used in Identification Schemes. In *Crypto '94*, LNCS 839, pages 202–215. Springer-Verlag, Berlin, 1994.
16. D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading MA, 1989.
17. O. Goldreich, S. Micali, and A. Wigderson. How to Prove All NP Statements in Zero-Knowledge and a Methodology of Cryptographic Protocol Design. In *Crypto '86*, LNCS 263, pages 171–185. Springer-Verlag, Berlin, 1987.
18. S. Goldwasser, S. Micali, and C. Rackoff. The Knowledge Complexity of Interactive Proof Systems. In *Proc. of the 17th STOC*, pages 291–304. ACM Press, New York, 1985.
19. L. C. Guillou and J.-J. Quisquater. A Practical Zero-Knowledge Protocol Fitted to Security Microprocessor Minimizing Both Transmission and Memory. In *Eurocrypt '88*, LNCS 330, pages 123–128. Springer-Verlag, Berlin, 1988.
20. S. Halevi and S. Micali. Practical and Provably-Secure Commitment Schemes from Collision-Free Hashing. In *Crypto '96*, LNCS 1109, pages 201–215. Springer-Verlag, Berlin, 1996.
21. J. H. Holland. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, 1975.
22. H. J. Knobloch. A Smart Card Implementation of the Fiat-Shamir Identification Scheme. In *Eurocrypt '88*, LNCS 330, pages 87–95. Springer-Verlag, Berlin, 1988.
23. L. Knudsen and W. Meier. Cryptanalysis of an Identification Scheme Based on the Permuted Perceptron Problem. In *Eurocrypt '99*, LNCS 1592, pages 363–374. Springer-Verlag, Berlin, 1999.

24. D. E. Knuth. *The Art of Computer Programming*, volume 2. Addison–Wesley, London, 1969.
25. M. Luby and Ch. Rackoff. How to Construct Pseudorandom Permutations from Pseudorandom Functions. *SIAM Journal of Computing*, 17(2):373–386, 1988.
26. NIST. *Secure Hash Standard* (SHS). Federal Information Processing Standards PUBlication 180–1, April 1995.
27. K. Ohta and T. Okamoto. A Modification of the Fiat-Shamir Scheme. In *Crypto '88*, LNCS 403, pages 232–243. Springer-Verlag, Berlin, 1989.
28. H. Ong and C.P. Schnorr. Fast Signature Generation with a Fiat-Shamir-Like Scheme. In *Eurocrypt '90*, LNCS 473, pages 432–440. Springer-Verlag, Berlin, 1991.
29. C. Papadimitriou and M. Yannakakis. Optimization, Approximation, and Complexity Classes. *Journal of Computer and Systems Sciences*, 43:425–440, 1991.
30. J. Patarin and P. Chauvaud. Improved Agorithms for the Permuted Kernel Problem. In *Crypto '93*, LNCS 773, pages 391–402. Springer-Verlag, Berlin, 1994.
31. D. Pointcheval. Neural Networks and their Cryptographic Applications. In *Eurocode '94*, pages 183–193. INRIA, 1994.
32. D. Pointcheval. A New Identification Scheme Based on the Perceptrons Problem. In *Eurocrypt '95*, LNCS 921, pages 319–328. Springer-Verlag, Berlin, 1995.
33. D. Pointcheval. The Composite Discrete Logarithm and Secure Authentication. In *PKC '00*, LNCS, pages 113–128. Springer-Verlag, Berlin, 2000.
34. G. Poupard. A Realistic Security Analysis of Identification Schemes based on Combinatorial Problems. *European Transactions on Telecommunications*, 8(5):471–480, September 1997.
35. G. Poupard and J. Stern. Security Analysis of a Practical "on the fly" Authentication and Signature Generation. In *Eurocrypt '98*, LNCS 1403, pages 422–436. Springer-Verlag, Berlin, 1998.
36. G. Poupard and J. Stern. On The Fly Signatures based on Factoring. In *Proceedings of 6th ACM-CCS*, pages 37–45. ACM press, 1999.
37. G. Poupard and J. Stern. Short Proofs of Knowledge for Factoring. In *PKC 2000*, LNCS 1751, pages 147–166. Springer-Verlag, 2000.
38. B. Preneel. *Analysis and Design of Cryptographic Hash Functions*. PhD thesis, Katholieke Universiteit Leuven, Departement Elektrotechniek, January 1993.
39. R. Rivest. The MD5 Message-Digest Algorithm. RFC 1321, The Internet Engineering Task Force, April 1992.
40. R. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.
41. C. P. Schnorr. Efficient Identification and Signatures for Smart Cards. In *Crypto '89*, LNCS 435, pages 235–251. Springer-Verlag, Berlin, 1990.
42. A. Shamir. An Efficient Identification Scheme Based on Permuted Kernels. In *Crypto '89*, LNCS 435, pages 606–609. Springer-Verlag, Berlin, 1990.
43. V. Shoup. On The Security of a Practical Identification Scheme. In *Eurocrypt '96*, LNCS 1070, pages 344–353. Springer-Verlag, Berlin, 1996.
44. J. Stern. A New Identification Scheme Based on Syndrome Decoding. In *Crypto '93*, LNCS 773, pages 13–21. Springer-Verlag, Berlin, 1994.
45. J. Stern. Designing Identification Schemes with Keys of Short Size. In *Crypto '94*, LNCS 839, pages 164–173. Springer-Verlag, Berlin, 1994.
46. J. Stern. A New Paradigm for Public-Key Identification. *IEEE Transaction on Information Theory*, IT–42:1757–1768, 1996.