A New Identification Scheme
Based on
the Perceptron Problem

David POINTCHEVAL
David.Pointcheval@ens.fr

Laboratoire d’Informatique, École Normale Supérieure
45, rue d’Ulm, F-75230 PARIS Cedex 05
E-mail: David.Pointcheval@ens.fr

Abstract. Identification is a useful cryptographic tool. Since zero-knowledge theory
appeared [3], several interactive identification schemes have been proposed (in particu-
lar Fiat-Shamir [2] and its variants [4, 6, 5], Schnorr [9]). These identifications are based
on number theoretical problems. More recently, new schemes appeared with the pecu-
liarity that they are more efficient from the computational point of view and that their
security is based on $\mathcal{NP}$-complete problems: PKP (Permuted Kernels Problem) [10],
SD (Syndrome Decoding) [12] and CLE (Constrained Linear Equations) [13].
We present a new $\mathcal{NP}$-complete linear problem which comes from learning machines:
the Perceptron Problem. We have some constraints, $m$ vectors $X_i$ of $\{-1, +1\}^n$, and
we want to find a vector $V$ of $\{-1, +1\}^n$ such that $X_i \cdot V \geq 0$ for all $i$.
Next, we provide some zero-knowledge interactive identification protocols based on this
problem, with an evaluation of their security. Eventually, those protocols are well suited
for smart card applications.

1 Introduction

An interactive identification protocol involves two persons Alice and Bob. Alice
wants to prove interactively that she is really Alice. She has a public key which
everybody knows, and a secret key associated to her public key. She is the only
one who knows the secret key, and nobody can compute it. To prove her identity,
Alice proves that she knows a secret key associated to her public key. In general,
the public key is a problem, a difficult problem, and the secret key is a solution
of this problem.

Recently, the zero-knowledge theory showed that we can prove the knowledge
of a solution of a problem without revealing anything about this solution. The
verifier learns nothing but the conviction that the prover knows a solution.

The first efficient zero-knowledge protocols were based on number theoretical
problems (Fiat-Shamir [2] and its variants [4, 6, 5], Schnorr [9]). They have two
major disadvantages:

- The hardness of the
3 Size of the problem

Secure values of $m$ and $n$ will be precised later with the efficiency of the attacks. But we will see that efficiency of the attacks will depend on the number of solutions of the instance.

So we will first evaluate the number of solutions of an average instance $(A, S)$ of the Permuted Perceptrons Problem of size $m \times n$ when we know there exists at least one solution $V$.

- On one hand, we can count the solutions for the Perceptrons Problem instance associated. Let $PP(m, n)$ be the average number of solutions for the Perceptrons Problem instance $A$ of size $m \times n$ (it is a big formula).

- On the other hand, we have to evaluate the probability to obtain a given multiset $S$ with the components of the product of $A$ and a given solution of $PP$: $P_{m, n, S}$

With every multiset $S$, this probability is less than the probability to obtain the multiset $\Sigma$ which elements have a Gaussian distribution ($i.e.$ $|\Sigma|_j = mp_{n, j}$, where $p_{n, j}$ is the probability for two $\varepsilon$-vectors of size $n$, $X$ and $Y$, to be such that $|X \cdot Y| = j$: $p_{n, j} = 2^{-n+1} \left( \frac{n}{n+j} \right) \approx \sqrt{\frac{8}{\pi n}} e^{-\frac{j^2}{4n}}$ when $n = j \mod 2$)

$$P_{m, n, S} \leq m! \prod_{j=1}^{n} \frac{p_{n, j}^{mp_{n, j}}}{(mp_{n, j})!}$$

Then the number of solutions for an average instance of PPP is less than $PP(m, n) \times P_{m, n, \Sigma}$.

Against the most efficient attack, we will need the smallest number of solutions. Then we will choose $m$ and $n$ such that $PP(m, n) \times P_{m, n, \Sigma} \leq 1$

As a consequence, we see that for interesting sizes ($100 < m < 200$), we can approximatively take $n \approx m + 16$.

4 Making the problem practical

4.1 How to get the keys?

For cryptographic purposes, we only use instances with a known solution. We also want all instances with at least one solution may appear. To get such an instance, we firstly choose an arbitrary $\varepsilon$-vector $V$, of size $n$, which will be the solution of the future instance, and an $\varepsilon$-matrix $A$ of size $m \times n$. We next modify it in the following way:

- If $(AV)_i < 0$, we replace the $i^{th}$ row of $A$ by its opposite value.
- If $(AV)_i \geq 0$, then we don’t change this row.

Then, we compute $S = \{(AV)_i | i = 1, \ldots, m\}$. Consequently, $(A, S)$ is an instance of the Permutated Perceptrons Problem, with $V$ as a solution. In addition, with this method, all the “good instances” may appear, with a probability which is proportional to the number of solutions of the PP instance.
4.2 Finite field

Also for cryptographic use, we need to bound the size of the numbers used in order to store them on a constant number of bits. So we will work in the finite field with \( p \) elements. If we bound the components of the product by a nonnegative odd integer \( t \) (i.e. if \( T \in \{1, 3, \ldots, t\}^m \)), we can prove that if \( n, p \) and \( t \) are such that \( 2p > n + t \) we have:

\[ AV = T \iff AV = T \mod p \]

5 Possible attacks

We tried several attacks against PP and PPP in order to evaluate the security of a possible protocol. But since there is no algebraic structure in those problems, no manipulation of the matrix will leave the problem unchanged (manipulation like Gaussian elimination, used in the past against PKP, CLE or any problem based on error correcting codes will not help here). So, it seems that only (more or less intelligent) exhaustive search or probabilistic attacks would succeed.

5.1 The majority vector

The first attack against PPP which comes to mind is the majority vector \( M \):

for all \( j \),

\[
M_j = \begin{cases} 
+1 & \text{if } \# \{ i | A_{i,j} = +1 \} > \frac{n}{2} \\
-1 & \text{otherwise}
\end{cases}
\]

**Theorem 4.** For an \( m \times n \)-instance constructed as shown below, with solution \( V \) and \( m \leq n \), the average Hamming distance between \( V \) and \( M \), is roughly

\[ n \cdot \left( \frac{1}{2} - \frac{1}{\pi} \sqrt{\frac{m}{n}} \right) \approx 0.2n. \]

Firstly, we can change 20% of the components of \( M \), and trying the products, but there are \( \binom{n}{0.2n} \) such possibilities.

We can already fix a bound for \( n \) (and \( m \)) to overtake the usual work factor of \( 2^{64} \): \( n \geq 95 \).

To improve this attack, we could arrange these changes beginning with components of \( M \) which values are litigious (i.e. \( \# \{ i | A_{i,j} = +1 \} \approx \frac{n}{2} \)). But surprisingly enough, some components which seem to be very good are false: on average, 80% of the components will have to be handled. So, this improvement doesn’t modify the bound on \( n \).

5.2 Simulated annealing

Because of the inefficiency of the previous attack, we tried the well-known probabilistic algorithm that comes from artificial intelligence, known as **simulated annealing** [11]. This attack tries to minimize a function, \( \text{Energy} \), defined on a finite metric space, in a probabilistic way. Simulated annealing algorithms are
an improvement of gradient descent algorithms. Whereas gradient descent algorithms can converge to a local minimum and stay there, simulated annealing algorithms try to go away with some small random perturbations. These perturbations which may be important at the beginning have to decrease to zero.

Such an algorithm can be efficient only if the Energy function is roughly “continuous” (the difference between the images of two neighbors is bounded by a small number). For this reason, simulated annealing doesn’t seem to be well suited for PPP because of the multiset, but it should be perfect for PP with

\[ E(V) = \frac{1}{2} \sum_{i=1}^{m} (|\langle AV \rangle_i| - \langle AV \rangle_i) \]

This algorithm turned out to be the most efficient. We have carried out many tests on square matrices \((m = n)\), and on some other sizes, and during a day, we can find a solution for any instance of PP which size is less than about 200.

Those attacks have been running for a few months and we never find a solution for PPP for sizes greater than 71.

If we suppose that each of those solutions can appear with the same probability, we have to repeat this attack about \(0.7 \times \#\{\text{solutions of PP}\}\) rounds, in order to find the good solution with probability 0.5.

Then, we can evaluate the work factor of such an attack with probability of success equal to 0.5:

<table>
<thead>
<tr>
<th>size</th>
<th>number of solutions PP</th>
<th>time for a solution (seconds)</th>
<th>time Solution Pr = 1/2 (seconds)</th>
<th>Work factor*</th>
</tr>
</thead>
<tbody>
<tr>
<td>101 × 117</td>
<td>(4.7 \times 10^9)</td>
<td>85</td>
<td>399.10^9</td>
<td>64</td>
</tr>
<tr>
<td>121 × 137</td>
<td>(8.7 \times 10^{10})</td>
<td>130</td>
<td>11.10^{12}</td>
<td>68</td>
</tr>
<tr>
<td>151 × 167</td>
<td>(3.7 \times 10^{12})</td>
<td>180</td>
<td>666.10^{12}</td>
<td>74</td>
</tr>
</tbody>
</table>

* work factor estimated using a 60-70 MIPS processor speed

Then, we can say that even small sizes are secure enough. In addition, whatever the probabilistic attack, it will not be able to differentiate the good solution of PPP from any solution of PP. So, even if we supposed a quick attack for PP (an \(\mathcal{NP}\)-complete problem) which would need only 1 second to find a solution for a 141 × 157-sized instance, the work factor would remain above \(2^{64}\).

6 Practical values

With those results, we can suggest \(m = 101\) and \(n = 117\) as a secure size of the problem. So, in the average instance, \(|S|_1 = 15\), \(|S|_3 = 14\), etc.

\[ \Pr_{\text{Instances}} [\text{Max } S > 33] < .3 \]

Then we can suppose that there is no greater number than 33, (i.e. \(t = 33\)). We must take \(p > 75\), and then we take \(p = 127\) (it will optimize the probability from a cheater to be rejected).
7 Protocols

Common data: some integers $p$, $n$, $t$ such that $2p > t + n$ and a collision-free, random hash function $H$. Let $A$ be a matrix of size $m \times n$, a Perceptrons Problem instance, with $V$ as a solution. Let $S$ be the multiset of the components of $AV$.

Public key: $(A, S)$
Secret key: $V$

The prover selects
• a random permutation $P$ over $\{0, \ldots, m - 1\}$ (to mix the rows of $A$.)
• a random signed permutation $Q$ over $\{0, \ldots, n - 1\}$ (to mix the columns of $A$, and to multiply them randomly by $+1$ or $-1$.)
• a random vector $W$ of $n$.

7.1 Three pass identification protocol ($3p \text{ zk}$)

1. The prover computes $A' = PAQ$, $V' = Q^{-1}V$, $R = W + V'$ and $h_0 = H(P|Q)$, $h_1 = H(W)$, $h_2 = H(R)$, $h_3 = H(A'W)$, $h_4 = H(A'R)$ and sends $(h_0, h_1, h_2, h_3, h_4)$ to the verifier.

2. The verifier randomly selects $c$ in $\{0, \ldots, 3\}$ and sends $c$ to the prover.

3. The prover sends:

   if $c = 0$: $(P, Q, W)$
   \[ h_0 = H(P|Q), h_1 = H(W), h_3 = H(PAQW). \]

   if $c = 1$: $(P, Q, R)$
   \[ h_0 = H(P|Q), h_2 = H(R), h_4 = H(PAQR). \]

   if $c = 2$: $(A'W, A'V')$
   \[ h_3 = H(A'W), h_4 = H(A'W + A'V') \]
   and $\{(A'V')_i\} = S$.

   if $c = 3$: $(W, V')$
   \[ h_1 = H(W), h_2 = H(W + V') \]
   and $V' \in \{-1, +1\}^n$.

7.2 Properties

Theorem 5. The $3p \text{ zk}$ protocol is an Interactive Proof System for $PPP$.

Lemma 6. Assume that some probabilistic polynomial-time adversary is accepted with probability greater than $\left(\frac{3}{4}\right)^r + \epsilon$ after $r$ rounds, then there exists a polynomial-time probabilistic machine which extracts the secret key $S$ from the public data or outputs collisions for the commitment function, with overwhelmingly probability.

Proof. Consider the tree $T(\omega)$ of all $4^k$ executions corresponding to all possible questions of the verifier over $k$ rounds when the adversary has a fixed random tape $\omega$.

\[ \alpha = \Pr_{\omega}[T(\omega) \text{ has a vertex with 4 sons}] \]

It is clear that $\alpha \geq \epsilon$, and by resetting the adversary $\frac{1}{\epsilon}$ times, one finds, with constant probability, an execution tree with a vertex having 4 sons. Repeating
again, this probability can be made very close to one.
A vertex with 4 sons corresponds to a situation where 5 commitments \( h_0, h_1, h_2, h_3 \) and \( h_4 \) have been made and where the adversary can provide answers to the 4 possible queries of the verifier.
Consider answers :

\[
\begin{align*}
H(P_0|Q_0) &= h_0 = H(P_1|Q_1) \\
H(W_0) &= h_1 = H(W_3) \quad H(P_3AQ_0W_0) = h_3 = H(Y_2) \\
H(R_1) &= h_2 = H(W_3 + V'_3) \quad H(P_1AQ_1R_1) = h_4 = H(Y_2 + Z_2)
\end{align*}
\]

Unless we have found a collision for the hash function \( H \), we can consider

\[
\begin{align*}
P &= P_0 = P_1 & R &= R_1 = W_3 + V'_3 = W + V' \\
Q &= Q_0 = Q_1 & Y &= Y_2 = P_0AQ_0W_0 = PAQW \\
W &= W_0 = W_3 & Y + Z = Y_2 + Z_2 = P_1AQ_1R_1 = PAQR
\end{align*}
\]
such that \( V' \in \{-1, +1\}^n \) and \( \{(Z_i)\} = S \).
so \( Y + Z = PAQR = PAQW + Z = PAQW + PAQV' \), then \( Z = PAQV' \).
Let \( V = QV' \) \((V \in \{-1, +1\}^n)\), then \( Z = PAQV \), Consequently \( \{(AV)_i\} = S \).

### 7.3 Five pass identification protocol (5p zk)

1. The prover computes \( A' = PAQ, V' = Q^{-1}V \)
   and \( h_0 = H(P|Q), h_1 = H(W|V'), h_2 = H(A'W|A'V') \)
   and sends \((h_0, h_1, h_2)\) to the verifier.
2. The verifier randomly selects \( k \) in \( \mathbb{F}_p^* \) and sends \( k \) to the prover.
3. The prover computes \( R = kW + V' \) and \( h_3 = H(R), h_4 = H(A'R) \)
   and sends \((h_3, h_4)\) to the verifier.
4. The verifier randomly selects \( c \in \{0, 1, 2\} \) and sends \( c \) to the prover.
5. The prover sends: 6. The verifier checks:

   if \( c = 0 : (P, Q, R) \) \quad \( h_0 = H(P|Q), h_3 = H(R), h_4 = H(PAQR) \)

   if \( c = 1 : (A'W, A'V') \) \quad \( h_2 = H(A'W|A'V'), h_4 = H(kA'W + A'V') \)
   \quad and \( \{(A'V'_i)\} = S \).

   if \( c = 2 : (W, V') \) \quad \( h_1 = H(W|V'), h_3 = H(kW + V') \)
   \quad and \( V' \in \{-1, +1\}^n \)

### 7.4 Properties

**Theorem 7.** The 5p zk protocol is an Interactive Proof System for PPP.

**Lemma 8.** Assume that some probabilistic polynomial-time adversary is accepted with probability greater than \( \left( \frac{2p - 1}{3(p - 1)} \right)^r + \epsilon \) after \( r \) rounds, then there exists a polynomial-time probabilistic machine which extracts the secret key \( S \) from the public data or outputs collisions for the commitment function, with overwhelming probability.

Using the idea of resettable simulation [3], in the random oracle model [1], it can be shown that both protocols are zero-knowledge. Alternatively, one has to assume specific statistical independance properties for the hash function.
7.5 Light versions

A light version (3p light and 5p light) of those protocols, which reduces the number of required rounds, can be designed.

Five pass identification protocol (5p light)
The initialization is the same as in the previous schemes.

1. The prover computes \( A_0 = PAQ \), \( V_0 = Q \)
   and \( h_0 = H(P|Q) \), \( h_1 = H(W|A'W|V'|A'V') \)
   and sends \((h_0, h_1)\) to the verifier.
2. The verifier randomly selects \( k \) in \( \mathbb{F}_p^* \) and sends \( k \) to the prover.
3. The prover computes \( R = kW + V' \) and \( h_2 = H(R|A'R) \)
   and sends \( h_2 \) to the verifier.
4. The verifier randomly selects \( c \) in \( \{0, 1\} \) and sends \( c \) to the prover.
5. The prover sends:
   6. The verifier checks:

   \[
   \begin{align*}
   \text{if } c = 0 & : (P, Q, R) \quad h_0 = H(P|Q), h_2 = H(R|PAQR) \\
   \text{if } c = 1 & : (W, V', A'W, A'V') \quad \{(A'V'_i)\} = S, \\
   & V' \in \{-1, +1\}^n \\
   & h_2 = H(X|Y) \text{ with} \\
   & X = kW + V' \\
   & Y = kA'W + 2T + U
   \end{align*}
   \]

These light protocols are no longer zero-knowledge. However, the information released appears quite small. In fact, in the case \( c = 1 \), the verifier learns two vectors and their images by \( A' \), then he can deduce something about \( A' \), and then about \( P \) and \( Q \). As he knows \( V' \), he theoretically learns a fraction of bit of \( V \). Since the permutations \( P \) and \( Q \) are different at each round, the given information seems unusable.

8 Performances

The performances of this scheme are similar to those of the already existing linear ones:

<table>
<thead>
<tr>
<th></th>
<th>SD Stern</th>
<th>CLE Stern</th>
<th>PKP Stern</th>
<th>Shamir</th>
<th>PPP 3p ZK</th>
<th>PPP 5p ZK</th>
</tr>
</thead>
<tbody>
<tr>
<td>matrix size</td>
<td>256 × 512</td>
<td>24 × 24</td>
<td>37 × 64</td>
<td>101 × 117</td>
<td></td>
<td></td>
</tr>
<tr>
<td>over the field</td>
<td>( \mathbb{F}_2 )</td>
<td>( \mathbb{F}_{16} )</td>
<td>( \mathbb{F}_{251} )</td>
<td>( \mathbb{F}_2 )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>best known attack</td>
<td>268</td>
<td>2^{52}</td>
<td>&gt; 2^{100}</td>
<td>2^{64}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>complexity</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of rounds</td>
<td>35</td>
<td>20</td>
<td>20</td>
<td>48</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>public key (bits)</td>
<td>256</td>
<td>80</td>
<td>296</td>
<td>144</td>
<td></td>
<td></td>
</tr>
<tr>
<td>secret key (bits)</td>
<td>512</td>
<td>80</td>
<td>384</td>
<td>117</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bits sent by round</td>
<td>954</td>
<td>824</td>
<td>832</td>
<td>896</td>
<td>1040</td>
<td></td>
</tr>
<tr>
<td>global transmission</td>
<td>4.08</td>
<td>2.01</td>
<td>2.03</td>
<td>5.25</td>
<td>4.44</td>
<td>(kbytes)</td>
</tr>
</tbody>
</table>
As we can see in the figure, with a secret key more secure than in some other schemes (work factor of the attack greater than $2^{64}$), and with a probability of $10^{-6}$ for a cheater to be accepted, an identification requires less than 4.5 kbytes of communication between the prover and the verifier (to be compared with the 2 kbytes for PKP and CLE, and the 4 kbytes for SD). And we can improve them by the use of hash trees.

Moreover, all the operations are no more than additions and subtractions between small integers (less than one byte) even modulo 2. They are well suited to a very minimal environment of 8-bit processors.

If we use a common matrix $M$, stored in the seed of a pseudo-random generator, and if the keys are:

- secret key : a random $\varepsilon$-vector $V$ of size $n$ (less than 15 bytes)
- public key : the $\varepsilon$-vector $L$ such that $L_i(MV)_i \geq 0$
  and the multiset $S = \{L_i(MV)_i\}$ (18 bytes)

It is very few bytes if we compare them with PKP or SD. But we should not forget that as for PKP, SD and CLE, this scheme is not identity based. It means that public keys have to be certified by an authority.

They require only simple operations so the program is very small. Moreover, the size of the data (common data and keys) are tiny. As a consequence, little EEPROM is needed.

Very few temporary computations have to be stored so they require very little RAM.

9 Conclusion

We have defined a new identification scheme which is very easy to implement on every kind of smart card because of its very simple operations and the small size of the data. We welcome attacks from readers.

Acknowledgements

I would like to thank Louis Granboulan for the results about the majority vector, and Jacques Stern for fruitful discussions.

References


