ECOLE NORMALE SUPERIEURE

Les Réseaux de Neurones et leurs Applications Cryptographiques

David POINTCHEVAL

LIENS - 95 - 2

Département de Mathématiques et Informatique

CNRS 1327

Les Réseaux de Neurones et leurs Applications Cryptographiques

David POINTCHEVAL

LIENS - 95 - 2

Février 95

Laboratoire d'Informatique de l'Ecole Normale Supérieure
45 rue d'Ulm 75230 PARIS Cedex 05

Tel : (33)(1) 44 32 00 00

Adresse électronique : .. @dmi.ens.fr

Les Réseaux de Neurones et leurs Applications Cryptographiques

David Pointcheval
David.Pointcheval@ens.fr

Groupe de Recherche En Complexité et Cryptographie
Laboratoire d'Informatique de l'École Normale Supérieure
45, rue d'Ulm
75230 Paris Cedex 05

6 février 1995

Résumé

L'identification est un outil cryptographique très important. Avec la théorie du *zero-knowledge* [7], de nombreux schémas interactifs d'identification ont été proposés (en particulier, Fiat-Shamir [4] et ses variantes [13, 10, 9], Schnorr [14], etc). Ces schémas sont basés sur des problèmes de théorie des nombres. Plus récemment, de nouveaux schémas sont apparus avec la particularité qu'ils mettent en jeu des petits entiers, et dont la sécurité dépend de problèmes $\mathcal{NP}$-complets : PKP (Permuted Kernels Problem) [15], SD (Syndrome Decoding) [18] ou CLE (Constrained Linear Equations) [17].

Nous allons présenter un nouveau problème $\mathcal{NP}$-complet qui provient des réseaux de neurones : le Problème des Perceptrons. Nous avons des contraintes, m vecteurs X^i de taille n à coordonnées dans $\{-1, +1\}$, et nous cherchons un vecteur V de $\{-1, +1\}^n$ tel que pour tout i, $X^i \cdot V \geq 0$.

Nous présenterons ensuite des protocoles interactifs d'identification *zero-knowledge* basés sur ce problème, avec une analyse de la sécurité. Finalement, ces protocoles semblent tout à fait adaptés pour un usage sur un système à faibles ressources, telles les cartes à puces.

Mots clés : cryptographie, identification, zero-knowledge, problème des perceptrons

Introduction

Un protocole interactif d'identification met en scène deux personnes : Alice et Bob. Alice veut prouver interactivement son identité à Bob. Pour cela, Alice possède une clé publique, et une clé secrète, comme tout utilisateur. Cette clé publique est connue de tous, et il n'y a aucun doute sur l'identité de son propriétaire. En revanche, seule Alice connaît une clé secrète associée. Et pour tout autre individu, deviner ou calculer la clé secrète à partir de la clé publique d'Alice est impossible. Ainsi, pour prouver son identité, Alice prouve sa connaissance d'une clé secrète associée à sa clé publique. La théorie du *zero-knowledge* a montré qu'Alice pouvait faire ceci sans rien révéler sur sa clé secrète. Les premiers protocoles *zero-knowledge* furent basés sur des problèmes de théorie de nombres (Fiat-Shamir [4] et ses variantes [13, 10, 9], Schnorr [14], etc). Ils ont deux inconvénients majeurs :

- la difficulté des problèmes utilisés (factorisation et logarithme discret) n'est pas prouvée. Et des algorithmes, ainsi que des ordinateurs, de plus en plus performants les menacent.
- les opérations nécessaires (multiplications et exponentiations modulaires) sont très coûteuses en temps et en puissance machine.

En revanche, depuis 1989, de nouveaux schémas sont apparus, qui reposent sur des problèmes $\mathcal{NP}$-complets, et qui ne nécessitent des opérations que sur des petits entiers, voire sur des bits : PKP (Permuted Kernels Problem) [15], SD (Syndrome Decoding) [18] ou CLE (Constrained Linear Equations) [17].

Voici un nouveau schéma linéaire basé sur le Problème des Perceptrons, un problème $\mathcal{NP}$-complet, qui semble parfaitement adapté à un usage sur carte à puce.

1 Problème des Perceptrons

1.1 Enoncé du problème

Le problème suivant apparaît en physique lors de l'étude des perceptrons d'Ising, ainsi qu'en intelligence artificielle et les réseaux de neurones. Nous le dénommerons *Problème des Perceptrons*.

Notation 1 *Nous appellerons ε-matrice (ou vecteur) une matrice (ou vecteur) dont les composantes sont $+1$ ou -1.*

Définition 1 *Problème de décision* ***PP***
Données : une ε-matrice A de taille $m \times n$.
Problème : Existe-t-il un ε-vecteur Y de taille n tel que $AY \geq 0$?

Définition 2 *Problème de recherche*
Données : une ε-matrice A de taille $m \times n$.
Problème : Trouver un ε-vecteur Y de taille n tel que $AY \geq 0$.

1.2 $\mathcal{NP}$-complétude

Théorème 1 *Le Problème des Perceptrons est $\mathcal{NP}$-complet*

PREUVE : Montrons donc que ce problème est dans $\mathcal{NP}$, et qu'il est $\mathcal{NP}$-dur.

(1) Tout d'abord, il est clair que ce problème est dans $\mathcal{NP}$.

(2) Ensuite, montrons que ce problème est $\mathcal{NP}$-dur. Pour cela, nous allons réduire de façon polynomiale une instance de 3-SAT à une instance de ce problème.
Soit $\mathcal{C}$ une instance de 3-SAT : des 3-clauses $C^1, \ldots, C^q$ sur les variables $x_1, \ldots, x_k$.

On construit, pour chaque clause, un ε-vecteur de taille $2k$; à une clause C^i, on associe un ε-vecteur X^i, tel que :

$$\begin{array}{llclclcl} \text{si } x_p \in C^i, & X^i_p & = & +1 & \text{et} & X^i_{p+k} & = & +1 \\ \text{si } \bar{x}_p \in C^i, & X^i_p & = & -1 & \text{et} & X^i_{p+k} & = & -1 \\ \text{sinon} & X^i_p & = & +1 & \text{et} & X^i_{p+k} & = & -1 \end{array}$$

On construit, pour une distribution de vérité, un ε-vecteur de taille $2k$; à une distribution de vérité D, on associe V tel que :

$$\begin{array}{llclclcl} \text{si } x_p \in D, & V_p & = & +1 & \text{et} & V_{p+k} & = & +1 \\ \text{si } \bar{x}_p \in D, & V_p & = & -1 & \text{et} & V_{p+k} & = & -1 \end{array}$$

On remarque que

$$\begin{array}{lclcl} C^i \text{ satisfaite sous } D & \Longleftrightarrow & X^i \cdot V & \in & \{-2, +2, +6\} \\ C^i \text{ non satisfaite sous } D & \Longleftrightarrow & X^i \cdot V & = & -6 \end{array}$$

Donc, C^i satisfaite sous $D \Longleftrightarrow X^i \cdot V \geq -2$ et V correspond à une distribution de vérité.

Or, V est associé à une distribution de vérité si pour tout p, $V_p = V_{p+k}$.

On construit donc des ε-vecteurs $Z^1, \ldots, Z^k$ comme suit :

$$\begin{array}{lrcrcrcr} \text{si } p = j, & Z_p^j & = & +1 & \text{et} & Z_{p+k}^j & = & -1, \\ \text{si } p \neq j, & Z_p^j & = & -1 & \text{et} & Z_{p+k}^j & = & +1. \end{array}$$

Alors,

$$(\forall p \in \{1, \ldots, k\})(V_p = V_{p+k}) \Longleftrightarrow (\forall j \in \{1, \ldots, k\})(Z^j \cdot V = 0)$$

On définit des ε-vecteurs de taille $2k+2$:

$$\begin{array}{rcll} \tilde{V} & = & (V, +1, +1), & \\ \tilde{X}^i & = & (X^i, +1, +1), & \text{pour tout } i, \\ \tilde{Z}^j & = & (Z^j, +1, -1), & \text{pour tout } j, \\ \tilde{Z'}^j & = & (Z^j, -1, +1), & \text{pour tout } j, \\ \tilde{Y}^j & = & -\tilde{Z}^j, & \text{pour tout } j, \\ \tilde{Y'}^j & = & -\tilde{Z'}^j, & \text{pour tout } j. \end{array}$$

Et alors, si I est un sous-ensemble de $\{1, \ldots, q\}$,

$$(\forall i \in I) \begin{array}{l} (C^i \text{ satisfaite sous } D, \\ \text{distribution de vérité} \\ \text{associée à } \tilde{V}) \end{array} \Longleftrightarrow \left\{ \begin{array}{lcll} \tilde{X}^i \cdot \tilde{V} & \geq & 0 & (\forall i \in I), \\ \tilde{Y}^j \cdot \tilde{V} & \geq & 0 & (\forall j \in \{1, \ldots, k\}), \\ \tilde{Y'}^j \cdot \tilde{V} & \geq & 0 & (\forall j \in \{1, \ldots, k\}), \\ \tilde{Z}^j \cdot \tilde{V} & \geq & 0 & (\forall j \in \{1, \ldots, k\}), \\ \tilde{Z'}^j \cdot \tilde{V} & \geq & 0 & (\forall j \in \{1, \ldots, k\}). \end{array} \right.$$

Considérons la matrice A dont les lignes sont les ε-vecteurs X^i pour $i \in \{1, \ldots, q\}$, Y^j, Y'^j, Z^j et Z'^j pour $j \in \{1, \ldots, k\}$.

Alors

$$(\forall i \in \{1, \ldots, q\}) \begin{array}{l} (C^i \text{ satisfaite sous } D, \\ \text{distribution de vérité} \\ \text{associée à } \tilde{V}) \end{array} \Longleftrightarrow AV \geq 0$$

On transforme donc le problème 3-Sat à q clauses sur k variables en un problème des perceptrons de taille $m \times n$ avec $m = q + 4k$ et $n = 2k + 2$.

$\square$

1.3 Approximation

1.3.1 Problème d'optimisation

Nous avons montré que ce problème des perceptrons était difficile à résoudre, mais peut-être un algorithme d'approximation nous permettrait-il de nous rapprocher suffisamment d'une solution. Et bien nous allons voir que ce problème est de plus difficile à approximer.

Le problème d'optimisation associé est le suivant (MAX-PP):

$$\max_{Y\ \varepsilon-vecteur} |\{i|(AY)_i \geq 0\}|$$

1.3.2 Définitions

Définition 3 (L-réduction [11]) *Soient deux problèmes d'optimisation Opt et Opt'. On dit que Opt se réduit linéairement à Opt' s'il existe deux algorithmes polynomiaux f et g et deux constantes $\alpha, \beta > 0$ tels que pour toute instance x de Opt :*

1. *L'algorithme f construit une instance x' de Opt' telle que $Opt'(x') \leq \alpha Opt(x)$.*

2. *Pour toute approximation y' de $Opt'(x')$, de coût $c'(y')$, l'algorithme g construit une approximation y de $Opt(x)$, de coût $c(y)$ telle que :*

$$|Opt(x) - c(y)| \leq \beta |Opt'(x') - c'(y')|.$$

Définition 4 (MAX-$\mathcal{SNP}$) *C'est la classe des problèmes L-réductibles à* MAX-3-SAT.

Définition 5 (MAX-$\mathcal{SNP}$-dur) *Un problème d'optimisation est* MAX-$\mathcal{SNP}$*-dur si le problème* MAX-3-SAT *s'y réduit linéairement.*

Proposition 1.1 ([1]) *Il existe une constante $\epsilon > 0$ telle que* MAX-3-SAT *ne soit pas approximable à un facteur $1+\epsilon$ près (sous réserve que $\mathcal{P} \neq \mathcal{NP}$).*
Plus précisément, MAX-3-SAT *n'est pas approximable à $\frac{74}{73}$ près, si $\mathcal{P} \neq \mathcal{NP}$ ($\epsilon = \frac{1}{73}$) [3]*

Proposition 1.2 MAX-3-SAT *est approximable à $\frac{1}{7}$ près*

Corollaire 1.1 *Propriétés :*

1. *Sous réserve que $\mathcal{P} \neq \mathcal{NP}$, pour tout problème d'optimisation* MAX-$\mathcal{SNP}$*-dur, il existe une constante pour laquelle ce problème n'est pas approximable.*

2. *Tout problème d'optimisation de* MAX-$\mathcal{SNP}$ *peut être approximé à une constante près.*

1.3.3 Résultats

Théorème 2 *Ce problème d'optimisation* MAX-PP *est* MAX-$\mathcal{SNP}$*-dur*

PREUVE : Réduisons une instance $\mathcal{C}$ de MAX-2-SAT en une instance A de MAX-PP selon une L-réduction :
Soit $\mathcal{C}$ une instance de MAX-2-SAT : des 2-clauses $C^1, \ldots, C^m$ sur les variables $x_1, \ldots, x_n$. A chaque clause $C_i = \{y_i, z_i\}$ (où $y_i \in \{x_{j_i}, \bar{x}_{j_i}\}$ et $z_i \in \{x_{k_i}, \bar{x}_{k_i}\}$), on associe les ε-vecteurs de taille $2n$:

$$
\begin{array}{llllllll}
X^i & : & X^i_p & = & +1 & \text{et} & X^i_{p+n} & = & -1 & \text{pour } 1 \leq p \leq n, p \neq j_i, k_i \\
 & & X^i_j & = & +1 & \text{et} & X^i_j & = & +1 & \text{si } x_j \in C_i \\
 & & X^i_j & = & -1 & \text{et} & X^i_j & = & -1 & \text{si } \bar{x}_j \in C_i \\
Y^i & : & Y^i_p & = & +1 & \text{et} & Y^i_{p+n} & = & -1 & \text{pour } 1 \leq p \leq n, p \neq j_i \\
 & & Y^i_{j_i} & = & -1 & \text{et} & Y^i_{j_i+n} & = & -1 & \\
Z^i & : & Z^i_p & = & +1 & \text{et} & Z^i_{p+n} & = & -1 & \text{pour } 1 \leq p \leq n, p \neq k_i \\
 & & Z^i_{k_i} & = & -1 & \text{et} & Y^i_{k_i+n} & = & -1 & \\
W^i & : & W^i_p & = & +1 & \text{et} & W^i_{p+n} & = & -1 &
\end{array}
$$

Puis,

$$
\begin{array}{lll}
\tilde{Y}^i & = & -Y^i \\
\tilde{Z}^i & = & -Z^i \\
\tilde{W}^i & = & -W^i
\end{array}
$$

Considérons la matrice A dont les lignes sont les ε-vecteurs X^i, Y^i, $\tilde{Y}^i$, Z^i, $\tilde{Z}^i$, W^i et $\tilde{W}^i$ pour $i = 1, \ldots, m$. A est alors une instance de MAX-PP. Le but étant de déterminer un ε-vecteur V dont le nombre de composantes positives de son produit avec A est maximal.

Soit une distribution de vérité, D, de l'instance de 2-SAT initiale, $\mathcal{C}$, qui optimise le nombre de clauses satisfaites : s. Alors en posant $V_i = V_{i+n} = +1$ si $x_i \in D$, ou $V_i = V_{i+n} = -1$ sinon, V rend $6m + s$ composantes du produit AV positives. Montrons que cette réduction est linéaire :
Tout d'abord, on sait qu'on peut rendre au moins une clause sur 2 vraie, donc $Opt(\mathcal{C}) \geq \frac{m}{2}$. En revanche, la matrice A est taille $2n \times 7m$, donc $Opt(A) \leq 7 \cdot m$. Alors

$$Opt(A) \leq 14 \cdot Opt(\mathcal{C}).$$

Ensuite, soient U, un ε-vecteur de coût $c'(V) = |\{i|(AU)_i \geq 0\}|$ et Δ, la distribution définie par :

$$
\begin{cases}
x_i \in \Delta & \text{si } V_i = +1 \\
\bar{x}_i \in \Delta & \text{si } V_i = -1
\end{cases}
$$

Cas 1 : $c'(U) \leq 6m$, $|Opt(A) - c'(V)| \geq 6m + s - c'(V) \geq s \geq |Opt(\mathcal{C}) - c(D)|$ où $c(D)$ est le coût de n'importe quelle distribution de vérité D de l'instance $\mathcal{C}$ (*i.e.* $c(D)$ est le nombre de clauses de $\mathcal{C}$ satisfaites avec D).
En particulier, $|Opt(A) - c'(U)| \geq |Opt(\mathcal{C}) - c(\Delta)|$

Cas 2 : $c'(U) = 6m + t$ où $t > 0$, alors posons $E = \{i | U_i = -U_{i+n}\}$. Définissons le vecteur suivant : $U'_{i+n} = U'_i = U_i$, pour tout $i \in E$ et $U'_i = U_i$ pour $i = \varepsilon n + j$ où $\varepsilon = \pm 1$ et $j \notin E$.

Ainsi, pour tout i :

- $W^i \cdot U' \geq 0$ et $\tilde{W}^i \cdot U' \geq 0$.
- si $j_i \in E$ ou $k_i \in E$, alors au moins une des quatre inégalités ($Y^i \cdot U \geq 0$, $\tilde{Y}^i \cdot U \geq 0$, $Z^i \cdot U \geq 0$ et $\tilde{Z}^i \cdot U \geq 0$) n'était pas vérifiée, et elles le sont toutes désormais avec U' : au moins une inégalité de gagnée.
  En revanche, on peut ne plus avoir $X_i \cdot U' \geq 0$: au plus une inégalité de perdue
  Soit, au pire, autant d'inégalités d'indice i vérifiées
- si $j_i, k_i \notin E$, alors les inégalités $Y^i \cdot U \geq 0$, $\tilde{Y}^i \cdot U \geq 0$, $Z^i \cdot U \geq 0$ et $\tilde{Z}^i \cdot U \geq 0$ étaient vérifiées et le restent avec U'.
  Quant à $X^i \cdot U \geq 0$, elle garde son état avec U'.

Alors au moins $6m+t$ inégalités sont satisfaites avec ce nouveau vecteur U'. Sachant que pour tout i,

$$\begin{array}{lll} Y^i \cdot U' \geq 0, & Z^i \cdot U' \geq 0, & W^i \cdot U' \geq 0, \\ \tilde{Y}^i \cdot U' \geq 0, & \tilde{Z}^i \cdot U' \geq 0, & \tilde{W}^i \cdot U' \geq 0, \end{array}$$

la distribution Δ, définie ci-dessus, satisfait au moins t clauses. Donc $c(\Delta) \geq t$.

$$|Opt(A) - c'(U)| = 6m + s - 6m - t = s - t \geq |Opt(\mathcal{C}) - c(\Delta)|$$

Dans tous les cas, la distribution de vérité Δ, construite en temps polynomial, vérifie :

$$|Opt(A) - c'(U)| \geq |Opt(\mathcal{C}) - c(\Delta)|$$

□

Théorème 3 *Ce problème d'optimisation est approximable à 2 près.*

PREUVE : En effet, choisissons un ε-vecteur, Z, de façon aléatoire.
Posons $\nu = |\{i | (AZ)_i \geq 0\}|$.

- si $\nu \geq \frac{m}{2}$, on pose $Y = Z$
- sinon, $Y = -Z$

Alors, Y approxime l'optimum à moins de 2 près. □

2 Problème des Perceptrons Permutés

Le problème des Perceptrons étant dans $\mathcal{NP}$, il admet une preuve *zero-knowledge* [6]. Étant de plus difficile à résoudre, il permettrait la conception d'un protocole d'identification interactif *zero-knowledge*. Cependant, afin d'obtenir un protocole plus simple, et, qui plus est, plus sûr, nous allons définir une variante de ce problème :

Définition 6 *Problème des Perceptrons Permutés* ***PPP***

Données : *une ε-matrice A de taille $m \times n$.*
un multi-ensemble S, de m entiers positifs.

Problème : *Existe-t-il un ε-vecteur Y de taille n*
tel que $\{\{(AY)_i | i = 1, \dots m\}\} = S$?

Théorème 4 *Ce problème des perceptrons permutés* ***PPP*** *est $\mathcal{NP}$-complet.*

PREUVE :

Lemme 4.1 **(([5]))** *Le problème* ONE-IN-THREE 3-SAT *est $\mathcal{NP}$-complet :*

Données : *Ensemble de variables, U, et liste de 3-clauses, C, sur les variables de U.*

Problème : *Existe-t-il une distribution de vérité pour les variables de U*
telle que toute clause c de C ait un et un seul littéral de vrai ?

En effectuant les mêmes transformations que lors de la réduction de 3-SAT à **PP**, on obtient une réduction de ONE-IN-THREE 3-SAT à **PPP** :

$$\begin{array}{c}(\forall i \in \{1, \dots, q\}) \\ (C^i \text{ a exactement un littéral vrai sous } D, \\ \text{distribution de vérité associée à } \tilde{V})\end{array} \Longleftrightarrow \left\{ \begin{array}{lcll} \tilde{X}^i \cdot \tilde{V} & = & 0 & (\forall i \in \{1, \dots, q\}), \\ \tilde{Z}^j \cdot \tilde{V} & = & 0 & (\forall j \in \{1, \dots, k\}), \\ \tilde{Z'}^j \cdot \tilde{V} & = & 0 & (\forall j \in \{1, \dots, k\}). \end{array} \right.$$

Considérons la matrice A dont les lignes sont les ε-vecteurs $\tilde{X}^i$ pour $i \in \{1, \dots, q\}$, $\tilde{Z}^j$ et $\tilde{Z'}^j$ pour $j \in \{1, \dots, k\}$, ainsi que le multi-ensemble S composé de $2k + q$ fois l'élément nul. On a ainsi associé une instance de **PPP** de taille $m \times n$ où $n = 2k$ et $m = q + 2k$ à une instance de 3-SAT.

Ceci montre que même le sous-problème de **PPP** où tous les éléments de S sont identiques (tous nuls) est $\mathcal{NP}$-complet. $\square$

3 Applications cryptographiques

3.1 Construction d'une instance

Pour un usage cryptographique, il nous faut pouvoir construire une instance de **PPP** dont on connaisse une solution, tout en pouvant ainsi obtenir toute instance ayant une solution. Pour cela, on commence par tirer aléatoirement un ε-vecteur V, qui sera la future solution, ainsi qu'une ε-matrice A de taille $m \times n$. Ensuite, on modifie cette matrice :

- Si $(AV)_i < 0$, on remplace la i^{e} ligne de A par son opposé.
- Sinon, on laisse cette ligne inchangée

Enfin, on calcule $S = \{\{(AV)_i | i = 1, \dots, m\}\}$.

Alors, (A, S) est une instance de **PPP** admettant V pour solution. De plus, par cette construction, toute instance ayant une solution peut apparaître, avec une probabilité proportionnelle au nombre de solutions que l'instance de **PP** associée a. C'est-à-dire que plus une instance de **PP** a de solutions, plus elle a de chances d'apparaître.

3.2 Corps fini

Toujours pour une application cryptographique, il faut borner les tailles des nombres manipulés pour les stocker sur un nombre constant de bits. On ramène alors le problème dans un corps fini :
On considère

- le corps fini à p éléments, $\mathbb{F}_p$ (p impair).
- une ε-matrice A de taille $m \times n$, n impair.
- un vecteur T de taille m à composantes entières positives impaires majorées par $t < p$.

On cherche à remplacer le test $AY = T$ par $AY = T \bmod p$, c'est-à-dire qu'un ε-vecteur Y sera accepté si

$$(\forall i)(AY)_i \in \{kp + T_i | \text{ k entier relatif}\}.$$

Or, pour tout i, la i^{e} composante du produit est impaire, donc l'ensemble se restreint à

$$\{kp + T_i | \ k \text{ pair}\}.$$

Par conséquent, les cas à refuser sont $(AY)_i \leq -2p + T_i$ ou $(AY)_i \geq 2p + T_i$, Or, pour tout i et tout Y, $|(AY)_i| \geq n$.
Donc, la condition $2p > n + t$ entraîne l'équivalence :

$$AY = T \Longleftrightarrow AY = T \bmod p$$

3.3 Codage du problème

Tout d'abord, chaque individu choisit sa clé secrète V dans $\{-1,+1\}^n$.
Ensuite, pour les diverses ε-matrices, les individus peuvent utiliser une matrice M aléatoire commune à tous et la clé publique sera alors :

- $L \in \{-1,+1\}^m$ tel que pour tout j, $L_j(MV)_j > 0$
- $S = \{\{L_j(MV)_j | j = 1, \ldots, m\}\}$

Notation 2 *Moyenne(m,n,y) est le nombre moyen de d'éléments de S égaux à y pour une instance de taille $m \times n$:*

$$Moyenne(m,n,y) = \lceil \frac{m}{2^{n-1}} \binom{n}{\frac{y+n}{2}} \rfloor$$

Ainsi, le stockage des clés peut se faire de la façon suivante :

Clé secrète	:	ε-vecteur V de taille n	$\rightarrow n$ bits
Clé publique	:	ε-vecteur L de taille m	$\rightarrow m$ bits
		vecteur D de taille $\frac{t-1}{2}$ à composantes dans $\{-2^l+1, \ldots, 2^l-1\}$	$\rightarrow \frac{l(t-1)}{2}$ bits

La preuve consistera à montrer la connaissance d'un ε-vecteur X tel que

$$(\forall y \in \{1,3,\ldots,t\}) \qquad \#\{j | \mathcal{L}_j(MX)_j = y\} = D_{\frac{y-1}{2}} + \text{Moyenne}(m,n,y)$$

3.4 Dimensions à utiliser

Ainsi que nous allons le voir par la suite, la valeur de m et de n dépendra de l'efficacité des attaques. Cependant, on peut déjà chercher le nombre de solutions pour une instance moyenne de **PPP**. Car on verra par la suite que moins il y a de solutions, moins les attaques seront efficaces. Pour cela, il nous faut savoir :

- le nombre de solutions pour **PP**.
- la probabilité d'obtenir un multi-ensemble donné pour S.

3.4.1 Nombre de solutions pour PP

Lorsque l'on connaît l'existence d'au moins une solution, on peut évaluer, par une méthode combinatoire, le nombre total de solutions pour **PP** : Soient X et V deux ε-vecteurs tels que $X \cdot V = \alpha$, alors

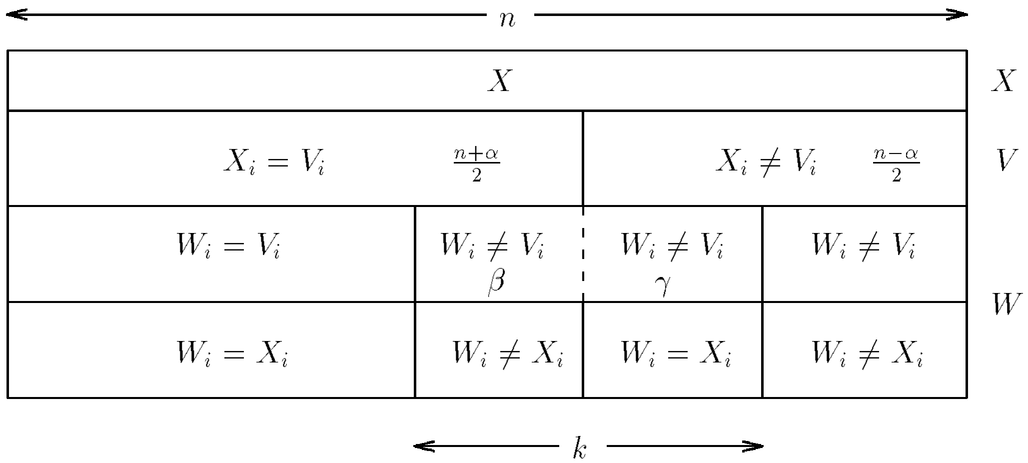
$$\left\{ \begin{array}{lcl} \#\{X_i = V_i\} & = & \dfrac{n+\alpha}{2} \\ \#\{X_i \neq V_i\} & = & \dfrac{n-\alpha}{2} \end{array} \right.$$

Pour tout ε-vecteur W tel que

$$\left\{ \begin{array}{lcl} d_H(W,V) & = & k \\ X \cdot W & = & \delta \end{array} \right., \text{et} \left\{ \begin{array}{lcl} \#\{W_i \neq V_i \text{ et } X_i = V_i\} & = & \beta \\ \#\{W_i \neq V_i \text{ et } X_i \neq V_i\} & = & \gamma \end{array} \right.,$$

on a

$$\begin{aligned} \beta+\gamma &= k \\ \frac{n+\alpha}{2}-\beta+\gamma &= \frac{n+\delta}{2} \end{aligned}$$



Alors,

$$\begin{aligned} \beta &= \frac{k}{2} - \frac{\delta-\alpha}{4} \\ \gamma &= \frac{k}{2} + \frac{\delta-\alpha}{4} \end{aligned} \qquad \text{avec} \qquad \begin{cases} 0 \leq \beta,\gamma \leq k \\ 0 \leq \beta \leq \dfrac{n+\alpha}{2} \\ 0 \leq \gamma \leq \dfrac{n-\alpha}{2} \end{cases}$$

Par suite, on a une contrainte sur δ :

$$-2\cdot\min\{k, n+\alpha-k\} \leq \delta-\alpha \leq 2\cdot\min\{k, n-\alpha-k\}$$

Considérons les ε-vecteurs W à distance paire de V (*i.e.* $k \in 2\mathbb{N}$):

$$\Pr_{\substack{W \\ d_H(V,W)=k}}[X\cdot W = \delta | X\cdot V = \alpha] = \frac{\binom{\frac{n+\alpha}{2}}{\beta}\binom{\frac{n-\alpha}{2}}{\gamma}}{\binom{n}{k}}$$

Pour tout $\alpha \geq 0$:

$$\Pr_{\substack{W \\ d_H(V,W)=k}}[X\cdot W \geq 0 | X\cdot V = \alpha] = \sum_{\substack{\delta\geq 0 \\ -\min\{k,n+\alpha-k\} \\ \leq \frac{\delta-\alpha}{2} \leq \\ \min\{k,n-\alpha-k\} \\ \delta-\alpha\in 4\mathbb{Z}}} \frac{\binom{\frac{n+\alpha}{2}}{\beta}\binom{\frac{n-\alpha}{2}}{\gamma}}{\binom{n}{k}}$$

Ainsi, le nombre de solutions à distance k, paire, de V, tel que $AV = Y$, est égal à

$$\begin{aligned} N(m,n,k,Y) &= \binom{n}{k} \Pr_{\substack{W \\ d_H(V,W)=k}} [AW \geq 0] \\ &= \binom{n}{k}^{-(m-1)} \prod_{i=1}^{m} \sum_{\substack{\delta \geq 0 \\ -\min\{k,n+Y_i-k\} \\ \leq \frac{\delta - Y_i}{2} \leq \\ \min\{k,n-Y_i-k\} \\ \delta - Y_i \in 4\mathbb{Z}}} \binom{\frac{n+Y_i}{2}}{\frac{2k-\delta+Y_i}{4}} \binom{\frac{n-Y_i}{2}}{\frac{2k+\delta-Y_i}{4}} \end{aligned}$$

Et alors, le nombre de solutions à distance paire de V, tel que $AV = Y$, est égal à

$$N(m,n,Y) = \sum_{\substack{0 \leq k \leq n \\ 2|k}} \binom{n}{k}^{-(m-1)} \prod_{i=1}^{m} \sum_{\substack{\delta \geq 0 \\ -\min\{k,n+Y_i-k\} \\ \leq \frac{\delta - Y_i}{2} \leq \\ \min\{k,n-Y_i-k\} \\ \delta - Y_i \in 4\mathbb{Z}}} \binom{\frac{n+Y_i}{2}}{\frac{2k-\delta+Y_i}{4}} \binom{\frac{n-Y_i}{2}}{\frac{2k+\delta-Y_i}{4}}$$

Par conséquent, on peut évaluer le nombre moyen de solutions à distance paire de V, pour une instance de taille $m \times n$. En effet, le vecteur $Y = AV$ suit une distribution Gaussienne

$$\left(i.e. \Pr_{\alpha}[Y_i = \alpha] = 2^{-n+1} \binom{n}{\frac{n+\alpha}{2}} \simeq \sqrt{\frac{8}{\pi n}} e^{-\frac{\alpha^2}{2n}} \right),$$

Alors, le nombre moyen de solutions pour une instance du Problème des Perceptrons de taille $m \times n$ est environ

$$\tilde{N}(m,n) = 2 \times \sum_{\substack{0 \leq k \leq n \\ 2|k}} \binom{n}{k}^{-(m-1)} \prod_{\alpha=0}^{\frac{n}{2}-1} \left[\sum_{\substack{\delta \geq 0 \\ -\min\{k,n+2\alpha+1-k\} \\ \leq \frac{\delta - 2\alpha - 1}{2} \leq \\ \min\{k,n-2\alpha-1-k\} \\ \delta - 2\alpha - 1 \in 4\mathbb{Z}}} \binom{\frac{n+2\alpha+1}{2}}{\frac{2k-\delta+2\alpha+1}{4}} \binom{\frac{n-2\alpha-1}{2}}{\frac{2k+\delta-2\alpha-1}{4}} \right]^{2m \frac{\binom{n}{\frac{n+2\alpha+1}{2}}}{2^n}}$$

3.4.2 Probabilité pour chaque multi-ensemble

Soit S^m un multi-ensemble à m éléments,

Notation 3 *On notera P_{m,n,S^m} la probabilité pour un vecteur Y qui vérifie $AY \geq 0$ de satisfaire $\{\{(AY)_i\}\} = S^m$.*

Notation 4 *On notera $|S^m|_j$ le nombre d'éléments de S^m égaux à j.*

Alors

$$P_{m,n,S^m} = \frac{m!}{|S^m|_1! \ldots |S^m|_n!} \prod_{j=1}^{j=n} p_{n,j}^{|S^m|_j} = m! \prod_{j=1}^{j=n} \frac{p_{n,j}^{|S^m|_j}}{|S^m|_j!}$$

où

$$p_{n,j} = \Pr_{X,Y}[|X \cdot Y| = j] \simeq \sqrt{\frac{8}{\pi n}} e^{-\frac{j^2}{2n}} \text{ si } j \text{ est impair, et 0 sinon.}$$

Il est clair que la probabilité P_{m,n,S^m} est maximale si S^m a une répartition Gaussienne Σ^m (*i.e.* $|\Sigma^m|_i = mp_{n,i}$).
Ainsi, pour toute répartition S^m,

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

3.4.3 Conclusion

Avec ces évaluations, on peut dire que le nombre de solutions, suivant un multi-ensemble donné S^m est à peu près

$$\tilde{N}(m,n) \times P_{m,n,S^m} \leq \tilde{N}(m,n) \times P_{m,n,\Sigma^m}$$

Ainsi, si on ne veut qu'une solution, il suffit de choisir m et n tels que

$$\tilde{N}(m,n) \times P_{m,n,\Sigma^m} < 2$$

i.e.

$$2 \times \sum_{\substack{0 \leq k \leq n \\ 2 | k}} \binom{n}{k}^{-(m-1)} \prod_{\alpha=0}^{\frac{n}{2}-1} \left[\sum_{\substack{\delta \geq 0 \\ -\min\{k, n+2\alpha+1-k\} \\ \leq \frac{\delta-2\alpha-1}{2} \leq \\ \min\{k, n-2\alpha-1-k\} \\ \delta-2\alpha-1 \in 4\mathbb{Z}}} \binom{\frac{n+2\alpha+1}{2}}{\frac{2k-\delta+2\alpha+1}{4}} \binom{\frac{n-2\alpha-1}{2}}{\frac{2k+\delta-2\alpha-1}{4}} \right]^{2m \frac{\binom{n}{\frac{n+2\alpha+1}{2}}}{2^n}} \times m! \prod_{j=1}^{j=n} \frac{p_{n,j}^{mp_{n,j}}}{(mp_{n,j})!} < 2$$

Par conséquent, les dimensions possibles sont les suivantes :

m	n maximal	Nombre de solutions pour une instance moyenne de **PP**	Probabilité Σ^m	Nombre de solutions pour une instance moyenne de **PPP**
101	117	9.4×10^{9}	1.0×10^{-10}	1
111	129	1.0×10^{11}	1.4×10^{-11}	1.4
121	137	1.6×10^{11}	6.0×10^{-12}	1
131	149	$1,4 \times 10^{12}$	7.0×10^{-13}	1
141	157	2.6×10^{12}	3.5×10^{-13}	1
151	167	7.4×10^{12}	1.4×10^{-13}	1
161	177	5.2×10^{13}	2.2×10^{-14}	1
171	185	6.2×10^{13}	5.4×10^{-15}	1

En particulier, on remarque que les bonnes tailles sont de la forme $n \approx m + 16$.

4 Attaques

Diverses attaques ont été tentées pour mettre à l'épreuve la sécurité de ce problème. Cependant, en raison de l'absence de structure algébrique, aucune manipulation de la matrice ne laissera le problème inchangé (telle l'élimination gaussienne contre PKP, CLE ou tout problème sur les codes correcteurs d'erreurs). Ainsi, il semble que seules des recherches exhaustives (plus ou moins subtiles) ou des attaques probabilistes auront une chance de fonctionner.

4.1 Vecteur majorité

Le premier vecteur approché qui vient à l'esprit est le *vecteur majorité* M :
pour tout j, $\begin{cases} M_j = +1 & \text{si } |\{i|A_{i,j} = +1\}| > \frac{n}{2} \\ M_j = -1 & \text{sinon} \end{cases}$
On va étudier le cas où $m = 2q + 1$ et $n = 2p + 1$.

Théorème 5 *Pour une instance fabriquée, de solution V, la distance de Hamming entre V et M, en moyenne, est de l'ordre de* $n \cdot (\frac{1}{2} - \frac{1}{\pi}\sqrt{\frac{m}{n}})$.

PREUVE : Soit une ε-matrice de taille $m \times n$ A, instance de solution V fabriquée comme indiqué ci-contre.
Si on tire V et A_i de taille n au hasard, la probabilité que r éléments correspondent est

$$\Pr[A_i \cap V = r] = 2^{-n} \binom{n}{r},$$

donc après remplacement éventuel de A_i par $-A_i$, cette probabilité est

$$\Pr = 2^{-(n-1)} \binom{n}{r}.$$

Le nombre moyen d'éléments correspondants est donc

$$2^{-(n-1)} \sum_{2r>n} r \binom{n}{r}.$$

La probabilité qu'un élément fixé corresponde est

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

La probabilité que, pour un élément fixé, plus de la moitié des A_j corresponde à V est alors

$$\begin{aligned} G_{m,n} &= \sum_{2s>m} \binom{m}{s} (1 - F_n)^{m-s} {F_n}^s \\ &\simeq \frac{1}{2^m} \sum_{s=q+1}^{m} \binom{m}{s} + \frac{1}{2^m} \sum_{s=0}^{q} \binom{m}{s} \left[(1 + \frac{1}{\sqrt{\pi p}})^{m-s} (1 - \frac{1}{\sqrt{\pi p}})^s - 1\right] \end{aligned}$$

Simplifions les formules :

$$G_{m,n} \simeq \frac{1}{2} + \frac{1}{2^m} \sum_{s=0}^{q} \binom{m}{s} \frac{m-2s}{\sqrt{\pi p}} \simeq \frac{1}{2} + \frac{1}{\pi}\sqrt{\frac{m}{2p}}$$

D'où

$$G_{m,n} \simeq \frac{1}{2} + \frac{1}{\pi}\sqrt{\frac{m}{n}} \simeq 0.8$$

Par conséquent, la distance de Hamming entre V et M est de l'ordre de $(\frac{1}{2} - \frac{1}{\pi}\sqrt{\frac{m}{n}}) \cdot n \simeq 0.2 \cdot n$ $\square$

Une première attaque consiste donc à changer 20 % des composantes de M, et d'essayer le produit. Mais il y a $\binom{n}{0.20n}$ manières de choisir les composantes de M à changer. Cette première attaque nous oblige à prendre $n \geq 95$, pour amener la recherche à une complexité supérieure à 2^{64}.

Pour affiner cette attaque, on peut choisir en priorité les composantes de M dont les valeurs sont litigieuses (celles pour lesquelles $|\{i|(A_{i,j} = +1\}|$ proche de $\frac{n}{2}$). Mais de façon surprenante, des composantes de majorité écrasante peuvent être erronées : en moyenne, 80% des composantes devront être manipulées. Cette amélioration n'élève pas la limite en taille.

4.2 Faiblesse

On peut utiliser la faiblesse du problème **PPP** par rapport à **PP**. C'est-à-dire la connaissance du multi-ensemble que forment les composantes du produit. Cependant, il est encore trop coûteux d'essayer toutes les permutations possibles (environ $m!$), mais le nombre peut être sensiblement réduit en regroupant :

$$\begin{cases} \mathcal{A}_1 & = & \{i|\text{un nombre pair de composantes de } A_i \text{ sont à } +1\} \\ \mathcal{A}_2 & = & \{i|\text{un nombre impair de composantes de } A_i \text{ sont à } +1\} \\ \mathcal{B}_1 & = & \{\{(AV)_i = 1 \bmod 4\}\} \\ \mathcal{B}_2 & = & \{\{(AV)_i = 3 \bmod 4\}\} \end{cases}$$

$\mathcal{B}_1$ et $\mathcal{B}_2$ étant vus comme des multi-ensembles.

$$\text{Alors} \begin{cases} \#\mathcal{A}_1 = \#\mathcal{B}_1, & \text{et} \quad \#\mathcal{A}_2 = \#\mathcal{B}_2 \qquad (1) \\ & \text{ou} \\ \#\mathcal{A}_1 = \#\mathcal{B}_2, & \text{et} \quad \#\mathcal{A}_2 = \#\mathcal{B}_1 \qquad (2) \end{cases}$$

Comme nous avons pris n impair, soit (1), soit (2) est vérifié.
Supposons que ce soit (1). Alors il y a beaucoup moins de permutations à essayer. Cependant, la complexité rend toujours les calculs inimaginables ($(\frac{m}{2})!^2$). En revanche, cette approche nous permet de déterminer la parité de $+1$ (ou de -1) du vecteur solution :
En effet, en changeant 2 par 2 les coordonnées de V, le résidu modulo 4 de $(AV)_i$ reste inchangé. Si $n = 4q+1$ et si le nombre de $+1$ dans A_i pour $i \in \mathcal{A}_1$ est $2p$, alors

$$(A(-1, \ldots, -1))_i = -2p + (n-2p) = 4(q-p) + 1 = 1 \bmod 4$$

Donc, la solution a le même nombre, modulo 2, de composantes égales à $+1$ que le vecteur $(-1, \ldots, -1)$, c'est-à-dire un nombre pair.

4.3 Le recuit simulé

Vue l'inefficacité des attaques précédentes, nous avons tenté l'attaque probabiliste bien connue, notamment en intelligence artificielle, du *recuit simulé* [16]. Cette attaque tente de minimiser, de façon probabiliste, une fonction d'énergie définie sur un espace métrique fini. L'algorithme de recuit simulé est une amélioration de la méthode du gradient. Tandis que la méthode du gradient se trouve bloquée lorsqu'elle atteint un minimum local, le recuit simulé essaie d'en sortir par des perturbations aléatoires. L'amplitude de ces perturbations est importante au début, puis tend vers zéro.
Comme pour la méthode du gradient, il faut que la fonction d'énergie ait une certaine continuité, ou régularité, c'est-à-dire qu'elle ne fasse pas des sauts aberrants d'un point à son voisin. En particulier, le recuit simulé semble tout à fait adapté à l'attaque de **PP**, mais ne permettra pas de résoudre directement **PPP**.

Algorithme
On définit tout d'abord la fonction d'énergie que l'on cherchera à minimiser. Dans notre cas, pour résoudre **PP**, on prend la fonction d'énergie suivante : $E(V) = \#\{i | X_i.V < 0\}$, et on cherche à l'annuler. Ensuite, on a besoin de définir un voisinage pour tout vecteur, ici $\mathit{Vois}(V) = \{X | d_H(V, X) = 1\}$. Enfin, la performance de l'algorithme dépendra des trois paramètres Θ_i, Θ_f (températures initiale et finale) et τ (taux de décroissance de la température) :

1. Choix d'un vecteur S aléatoire dans l'espace des solutions possibles.
2. On pose $\Theta \longleftarrow \Theta_i$ (température initiale).
3. Choix de $V \in_R \mathit{Vois}(S)$
4. $\Delta \longleftarrow E(V) - E(S)$
5. si $\Delta > 0$ alors $p \longleftarrow \exp\left(-\frac{\Delta}{\Theta}\right)$, sinon $p \longleftarrow 1$.
6. $S \xleftarrow{\text{avec proba } p} V$
7. $\Theta \longleftarrow \Theta.\tau$
8. si $(E(S) > 0) \wedge (\Theta > \Theta_f)$ retourner en 3

Variantes
On peut améliorer cette attaque en utilisant la remarque précédente :

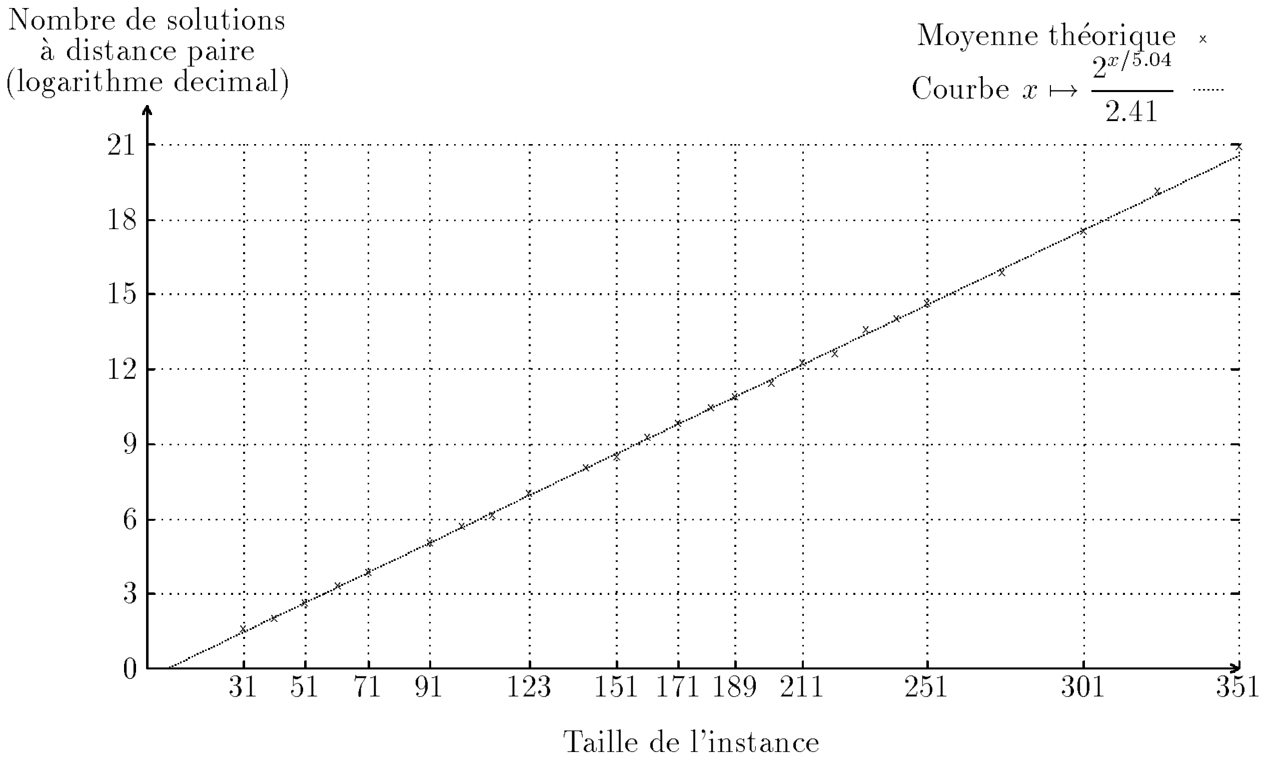
- On prend le vecteur initial S aléatoire parmi les vecteurs ayant la bonne parité de composantes à $+1$.
- On définit le voisinage $\mathit{Vois}(V) = \{X | d_H(X, V) = 2\}$. Ainsi, tous les vecteurs testés auront la bonne parité de $+1$.

Performances
Cette méthode optimisée se révèle être la plus efficace. Nous avons effectué de nombreux tests sur des matrices carrées ($m = n$), ainsi que sur diverses tailles, en moins d'une journée, on peut trouver une solution pour toute instance de **PP** de taille inférieure à environ 200.

m	n	temps (s)
31	31	40
51	51	60
71	71	75
101	117	85
123	123	105
121	137	130
151	151	1800
151	167	180
171	171	7200
189	189	36000

Depuis plusieurs mois que ces attaques tournent, jamais une solution pour **PPP**, pour des tailles supérieures à 71 (avec $n \approx m + 16$), n'a été trouvée.

Nombre de solutions sur des matrices carrées



Pour une matrice carrée, nous avons vu que le nombre théorique de solutions, à distance paire de la solution, pour **PP** est aux environs de $2^{n/5-2}$, et nos tests confirment cette évaluation. En effet, sur une instance carrée de taille 123, il faut calculer en moyenne 3000 solutions avant de retomber sur une solution déjà trouvée (une collision). En appliquant le paradoxe des anniversaires, on peut estimer qu'il y a environs 9 millions de solutions.

Conclusion

Pour trouver la bonne solution avec probabilité p, en supposant qu'on obtient, avec probabilité uniforme, toutes les solutions à distance paire de la bonne solution, il faut réitérer l'attaque :

$$\Pr[V \text{ pas trouvé en } k \text{ tours}] = \left(1 - \frac{1}{n}\right)^k$$

Donc

$$\Pr[V \text{ trouvé en } k \text{ itérations}] = 1 - \left(1 - \frac{1}{n}\right)^k \simeq 1 - e^{-\frac{k}{n}}$$

D'où

$$\Pr[V \text{ trouvé en } k \text{ itérations}] \geq p \Longleftrightarrow k \geq -\ln(1-p) \cdot n$$

Par conséquent, en réitérant $0.7 \times n$ fois l'attaque, on a une chance sur deux de trouver la bonne solution. On peut alors estimer le temps moyen sur notre machine pour avoir une chance sur deux de trouver la bonne solution :

Facteur de travail de l'attaque par recuit simulé

taille	nombre de solutions	temps pour une solution (secondes)	temps Solution Pr = 1/2 (secondes)		facteur* de travail 2^n opérations élémentaires
31×31	40	40	1.10^3	15 minutes	36
51×51	400	60	17.10^3	4 heures 45 minutes	40
71×71	7.500	75	400.10^3	4 jours 15 heures	44
101×117	$4.7\ 10^9$	85	399.10^9	12 mille ans	64
121×137	$8.7\ 10^{10}$	130	11.10^{12}	350 mille ans	68
123×123	$9.8\ 10^6$	105	720.10^6	22 ans et 10 mois	55
151×151	$306\ 10^6$	1800	385.10^9	12 mille ans	64
151×167	$3.7\ 10^{12}$	180	666.10^{12}	21 millions d'années	74
171×171	$6.7\ 10^9$	7200	34.10^{12}	1 millions d'années	70
189×189	$78\ 10^9$	36000	2.10^{15}	63 millions d'années	76

* facteur de travail estimé en utilisant le fait que la machine utilisée tourne aux environs de 60 à 70 MIPS

Ces diverses expériences nous permettent de proposer des tailles au problème pour une utilisation sûre en cryptographie. Le facteur de travail usuellement réclamé étant de 2^{64}, on pourra donc utiliser par exemple $m = 101$ et $n = 117$.

De plus, quelle que soit l'attaque probabiliste, elle ne pourra différencier la solution de **PPP** des nombreuses solutions de **PP**, alors même en supposant une attaque de l'ordre de quelques secondes contre **PP** (un problème $\mathcal{NP}$-complet) pour une instance de taille 141×157, le facteur de travail restera de l'ordre de 2^{64}.

5 Protocole d'identification à 3 passes

5.1 Protocole

Données communes à tout utilisateur :

p, n et t entiers tels que $2p > t + n$

H, une fonction de mise en gage aléatoire et résistante aux collisions.

Soit A une ε-matrice de taille $m \times n$, instance du Problème des Perceptrons avec V pour solution. Soit S le multi-ensemble formé des composantes du produit de A par V.
Clé publique : (A, S)
Clé secrète : V

Prouveur		Vérifieur
P, permutation aléatoire de $\{0, \ldots, m-1\}$ (Matrice qui échangera les lignes de A.) Q, permutation aléatoire signée de $\{0, \ldots, n-1\}$ (Matrice qui échangera les colonnes de A en les multipliant aléatoirement par $+1$ ou -1.) W, vecteur aléatoire de $\mathbb{F}_p^n$ $A' = PAQ$ $V' = Q^{-1}V$ $R = W + V'$ $h_0 = H(P\|Q)$ $h_1 = H(W)$, $h_3 = H(A'W)$, $h_2 = H(R)$, $h_4 = H(A'R)$.		
	$\xrightarrow{h_0, h_1, h_2, h_3, h_4}$	
	$\xleftarrow{c}$	$c \in_R \{0, 1, 2, 3\}$
Si $c = 0$	$\xrightarrow{P, Q, W}$	Vérifie $h_0 = H(P\|Q)$ $h_1 = H(W)$ $h_3 = H(PAQW)$
Si $c = 1$	$\xrightarrow{P, Q, R}$	Vérifie $h_0 = H(P\|Q)$ $h_2 = H(R)$ $h_4 = H(PAQR)$
Si $c = 2$	$\xrightarrow{A'W, A'V'}$	Vérifie $h_3 = H(A'W)$ $h_4 = H(A'W + A'V')$ $\{\{((A'V')_i\}\} = S$
Si $c = 3$	$\xrightarrow{W, V'}$	Vérifie $h_1 = H(W)$ $h_2 = H(W + V')$ $V' \in \{-1, +1\}^n$

5.2 Propriétés

Théorème 6 *Ce protocole est un système interactif de preuve pour* ***PPP****.*

PREUVE : Montrons donc la complétude puis la consistance.

Complétude: tout prouveur honnête (connaissant le secret V) saura répondre aux 4 questions, et donc convaincre le vérifieur avec probabilité 1.
Par conséquent,

$$Pr[\mathbf{A} \text{ convainque } \mathbf{B}] = 1$$

Consistance:

Lemme 6.1 *Supposons qu'il existe une machine de Turing Polynomiale Probabiliste adverse acceptée avec probabilité supérieure à* $\left(\frac{3}{4}\right)^k + \epsilon$ *après* k *itérations de ce protocole, alors il existe une machine de Turing Polynomiale Probabiliste qui extrait soit la clé secrète* V *associée aux données publiques, soit une collision pour la fonction de mise en gage, avec une probabilité écrasante.*

PREUVE : Considérons l'arbre d'exécution $T(\omega)$ correspondant à toutes les questions du vérifieur sur k tours quand la machine de Turing adverse a un ruban d'aléa égal à ω.

$$\alpha = \Pr_{\omega}[T(\omega) \text{ a un sommet avec 4 fils}]$$

Il est clair que $\alpha \geq \epsilon$, donc en réinitialisant $\frac{1}{\epsilon}$ fois cette machine adverse, on trouve,avec une probabilité constante un arbre d'exécution dont un des sommets a 4 fils. En répétant, cette probabilité peut être rendue aussi proche de 1 que l'on veut.
Un sommet avec 4 fils correspond à une situation où 5 mises en gages h_0, h_1, h_2, h_3, h_4 ont été effectuées, et où l'adversaire peut répondre aux 4 questions du vérifieur.
Considérons les réponses :

$$\begin{array}{lllll}
H(P_0|Q_0) & = & h_0 & = & H(P_1|Q_1) \\
H(W_0) & = & h_1 & = & H(W_3) \\
H(R_1) & = & h_2 & = & H(W_3 + V_3') \\
H(P_0AQ_0W_0) & = & h_3 & = & H(Y_2) \\
H(P_1AQ_1R_1) & = & h_4 & = & H(Y_2 + Z_2)
\end{array}$$

A moins d'avoir trouvé une collision pour H, on peut considérer

$$\begin{array}{lllllll}
P & = & P_0 & = & P_1 & & \\
Q & = & Q_0 & = & Q_1 & & \\
W & = & W_0 & = & W_3 & & \\
R & = & R_1 & = & W_3 + V_3' & = & W + V' \\
Y & = & Y_2 & = & P_0AQ_0W_0 & = & PAQW \\
Y + Z & = & Y_2 + Z_2 & = & P_1AQ_1R_1 & = & PAQR
\end{array}$$

tels que $V' \in \{-1, +1\}^n$ et $\{\{Z_i\}\} = S$.
Alors $Y + Z = PAQR = PAQW + Z = PAQW + PAQV'$, donc $Z = PAQV'$.
En posant $V = QV'$ (on a toujours $V \in \{-1, +1\}^n$), alors $Z = PAV$, d'où

$$\{\{(AV)_i\}\} = S.$$

Par conséquent, on a résolu le problème. □

$$Pr[\bar{\mathbf{A}} \text{ convainque } \mathbf{B}] \leq \left(\frac{3}{4}\right)^k$$

□

Théorème 7 *Dans le modèle de l'Oracle Aléatoire [2], ce protocole est un système interactif de preuve zero-knowledge.*

Preuve : Tout d'abord, on suppose que $m \leq n$ et que la matrice A est des rang m. Soit $\mathbf{Cr}$ la stratégie d'un vérifieur quelconque, ($i.e.$ $\mathbf{Cr}(h_0, h_1, h_2, h_3) \in \{0,1,2,3\}$). Voici la Machine de Turing Polynomiale Probabiliste M qui construit un ruban de communication indistingable d'un ruban créé au cours d'une réelle identification.

1. M choisit une question aléatoire $C \in \{0,1,2,3\}$

$C = 0$ M choisit aléatoirement P, Q et W
$h_0 = H(P,Q)$, $h_1 = H(W)$, $h_3 = H(PAQW)$, h_2 et h_4, chaînes aléatoires
Alors $H = \text{concat}(h_0, h_1, h_2, h_3, h_4)$ et Answer$=\text{concat}(P,Q,W)$.

$C = 1$ M choisit aléatoirement P, Q et R
$h_0 = H(P,Q)$, $h_2 = H(R)$, $h_4 = H(PAQR)$, h_1 et h_3, chaînes aléatoires
Alors $H = \text{concat}(h_0, h_1, h_2, h_3, h_4)$ et Answer$=\text{concat}(P,Q,R)$.

$C = 2$ M choisit un vecteur quelconque Y tel que $\{\{Y_i\}\} = S$, et un vecteur aléatoire X. Y est supposé être PAV et X, $PAQW$. Il est clair que Y a la même distribution que PAV, mais est-il vrai que la distribution de $PAQW$ est uniforme? Soit $Z \in \mathbb{F}_p^m$. On suppose PAV fixé, ce qui fixe P.

$$\begin{aligned}
\Pr_{Q,W}[PAQW = Z] &= \Pr_{Q,W}[AQW = P^{-1}Z] \\
&= \frac{\#\{(Q,W) | AQW = P^{-1}Z\}}{2^n n! p^n} \\
&= \frac{\sum_Q \#\{W | W = (PAQ)^{-1}\}}{2^n n! p^n} \\
\text{puisque rang}(A) = m, & \\
&= \frac{\sum_Q p^{n-m}}{2^n n! p^n} = \frac{2^n n! p^{n-m}}{2^n n! p^n} = \frac{1}{p^m}
\end{aligned}$$

$h_3 = H(X)$, $h_4 = H(X+Y)$, h_0, h_1 et h_2, chaînes aléatoires
Alors $H = \text{concat}(h_0, h_1, h_2, h_3, h_4)$ et Answer$=\text{concat}(X, X+Y)$.

$C = 3$ M choisit un vecteur aléatoire W, et un ε-vector aléatoire E.
$h_1 = H(W)$, $h_2 = H(W+E)$, h_0, h_3 and h_4, chaînes aléatoires
Alors $H = \text{concat}(h_0, h_1, h_2, h_3, h_4)$ et $R = \text{concat}(W, W+E)$.

2. M fait calculer $c = \mathbf{Cr}(H)$.

3. Si $c = C$ alors M écrit H, c et Answer, sinon M retourne à 1 (reset [7])

Alors M simule un ruban de communication indistingable d'un ruban d'une véritable identification de r tours en un nombre moyen de $4 \times r$ étapes. □

6 Protocole d'identification à 5 passes

6.1 Protocole

Données communes à tout utilisateur :

p, n et t entiers tels que $2p > t + n$

H, une fonction de mise en gage aléatoire et résistante aux collisions.

Soit A une ε-matrice de taille $m \times n$, instance du Problème des Perceptrons avec V pour solution. Soit S le multi-ensemble formé des composantes du produit de A par V.

Clé publique : $(A,\ S)$

Clé secrète : V

Prouveur		Vérifieur
P, permutation aléatoire de $\{0, \ldots, m-1\}$ (Matrice qui échangera les lignes de A.) Q, permutation aléatoire signée de $\{0, \ldots, n-1\}$ (Matrice qui échangera les colonnes de A en les multipliant aléatoirement par $+1$ ou -1.) W, vecteur aléatoire de $\mathbb{F}_p^n$		
$A' = PAQ$ $V' = Q^{-1}V$		
$h_0 = H(P, Q)$, $h_1 = H(W, V')$, $h_2 = H(A'W, A'V')$.	$\xrightarrow{h_0, h_1, h_2}$	
	$\xleftarrow{k}$	$k \in_R \mathbb{F}_p^\star$
$R = kW + V'$, $h_3 = H(R)$, $h_4 = H(A'R)$.	$\xrightarrow{h_3, h_4}$	
	$\xleftarrow{c}$	$c \in_R \{0, 1, 2\}$
Si $c = 0$	$\xrightarrow{P, Q, R}$	Vérifie $h_0 = H(P, Q)$ $h_3 = H(R)$ $h_4 = H(PAQR)$
Si $c = 1$	$\xrightarrow{A'W, A'V'}$	Vérifie $h_2 = H(A'W \vert A'V')$ $h_4 = H(kA'W + A'V')$ $\{\{(A'V')_i\}\} = S$
Si $c = 2$	$\xrightarrow{W, V'}$	Vérifie $h_1 = H(W \vert V')$ $h_3 = H(kW + V')$ $V' \in \{-1, +1\}^n$

6.2 Propriétés

Théorème 8 *Ce protocole est un système interactif de preuve pour **PPP**.*

PREUVE : Montrons donc la complétude puis la consistance.

Complétude: tout prouveur honnête (connaissant le secret V) saura répondre aux 3 questions, quelquesoit k, donc convaincre le vérifieur avec probabilité 1.
Consistance:

Lemme 8.1 *Supposons qu'il existe une machine de Turing Polynomiale Probabiliste adverse acceptée avec probabilité supérieure à* $\left(\frac{2p-1}{3(p-1)}\right)^k + \epsilon$ *après k itérations de ce protocole, alors il existe une machine de Turing Polynomiale Probabiliste qui extrait soit la clé secrète V associée aux données publiques, soit une collision pour la fonction de mise en gage, avec une probabilité écrasante.*

PREUVE : Considérons l'arbre d'exécution $T(\omega)$ correspondant à toutes les questions du vérifieur sur k tours quand la machine de Turing adverse a un ruban d'aléa égal à ω.

$$\alpha = \Pr_{\omega}[T(\omega) \text{ a un sommet avec au moins } 2p \text{ fils}]$$

Il est clair que $\alpha \geq \epsilon$, donc en réinitialisant $\frac{1}{\epsilon}$ fois cette machine adverse, on trouve,avec une probabilité constante un arbre d'exécution dont un des sommets a au moins $2p$ fils. En répétant, cette probabilité peut être rendue aussi proche de 1 que l'on veut.
Un sommet avec plus de $2p$ fils correspond à une situation où 3 mises en gages h_0, h_1, h_2 ayant été effectuée, il existe 2 valeurs de k, (parmi les $p-1$), pour lesquelles l'adversaire peut répondre aux 3 questions du vérifieur.
Considérons les réponses :

$$\begin{array}{lclcl}
H(P^1|Q^1) & = & h_0 & = & H(P^1|Q^1) \\
H(W^1|V'^1) & = & h_1 & = & H(W^2|V'^2) \\
H(Y^1|Z^1) & = & h_2 & = & H(Y^2|Z^2) \\
H(R^1) & = & h_3^1 & = & H(k^1W^1 + V'^1) \\
H(R^2) & = & h_3^2 & = & H(k^2W^2 + V'^2) \\
H(P^1AQ^1R^1) & = & h_4^1 & = & H(k^1Y^1 + Z^1) \\
H(P^2AQ^2R^2) & = & h_4^2 & = & H(k^2Y^2 + Z^2)
\end{array}$$

A moins d'avoir trouvé une collision pour H, on peut considérer

$$\begin{array}{lllll}
P & = & P^1 & = & P^2 \\
Q & = & Q^1 & = & Q^2
\end{array}
\qquad
\begin{array}{lllll}
W & = & W^1 & = & W^2 \\
V' & = & V'^1 & = & V'^2
\end{array}
\qquad
\begin{array}{lllll}
Y & = & Y^1 & = & Y^2 \\
Z & = & Z^1 & = & Z^2
\end{array}$$

$$\begin{array}{lclclcl}
R^1 & = & k^1W^1 + V'^1 & = & k^1W + V' & & \\
R^2 & = & k^2W^2 + V'^2 & = & k^2W + V' & & \\
PAQR^1 & = & P^1AQ^1R^1 & = & k^1Y^1 + Z^1 & = & k^1Y + Z \\
PAQR^2 & = & P^2AQ^2R^2 & = & k^1Y^2 + Z^2 & = & k^2Y + Z
\end{array}$$

$$\text{avec } V' \in \{-1, +1\} \text{ et } \{\{Z_i\}\} = S$$

Alors $R^1 - R^2 = (k^1 - k^2)W$ et $PAQ(R^1 - R^2) = (k^1 - k^2)Y$, et comme, $k^1 \neq k^2$, $PAQW = Y$. De plus, $PAQR^1 = k^1PAQW + PAQV' = k^1Y + PAQV' = k^1Y + Z$, donc $PAQV' = Z$. En posant $V = QV'$ (on a toujours $V \in \{-1, +1\}^n$), alors $Z = PAV$, d'où $\{\{(AV)_i\}\} = S$.
Par conséquent, on a résolu le problème. □

$$Pr[\bar{\mathbf{A}} \text{ convainque } \mathbf{B}] \leq \left(\frac{2}{3} + \frac{1}{3(p-1)}\right)^k \simeq \left(\frac{2}{3}\right)^k$$

□

Théorème 9 *Dans le modèle de l'Oracle Aléatoire [2], ce protocole est un système interactif de preuve zero-knowledge.*

PREUVE : Tout d'abord, on suppose que $m \leq n$ et que la matrice A est des rang m. Soit **Cr** la stratégie d'un vérifieur quelconque,
i.e. $\mathbf{Cr}(h_0, h_1, h_2) \in \mathbb{F}_p^\star$ et $\mathbf{Cr}(h_0, h_1, h_2, k, h_4, h_5) \in \{0, 1, 2\}$.
Voici la Machine de Turing Polynomiale Probabiliste M qui construit un ruban de communication indistingable d'un ruban créé au cours d'une réelle identification.

1. M choisit une question aléatoire $C \in \{0, 1, 2\}$

   $C = 0$ M choisit aléatoirement P, Q et W
   $h_0 = H(P, Q)$, h_1 and h_2, chaînes aléatoires.

   $C = 1$ M choisit un vecteur Y tel que $\{\{Y_i\}\} = S$, et un vecteur aléatoire X. Y est supposé être PAV et X, $PAQW$. Il est clair que Y a la même distribution que PAV, et nous avons déjà prouvé que la distribution de $PAQW$ est uniforme.
   $h_2 = H(X, Y)$, h_0 et h_1, chaînes aléatoires.

   $C = 2$ M choisit un vecteur aléatoire W, et un ε-vector aléatoire E.
   $h_1 = H(W, E)$, h_0 et h_2, chaînes aléatoires.

2. M fait calculer $k = \mathbf{Cr}(h_0, h_1, h_2)$.

3. Et alors

   $C = 0$ M choisit aléatoirement R et calcule $Y = PAQR$
   $h_3 = H(R)$, $h_4 = H(Y)$ et Answer= concat(P, Q, R).

   $C = 1$ M calcule $Z = Y + kX$
   $h_4 = H(Z)$, h_3, chaînes aléatoires et Answer= concat(X, Y).

   $C = 2$ M calcule $Y = E + kW$
   $h_3 = H(Y)$, h_4, chaînes aléatoires et Answer= concat(W, E).

4. M fait calculer $c = \mathbf{Cr}(h_0, h_1, h_2, k, h_3, h_4)$.

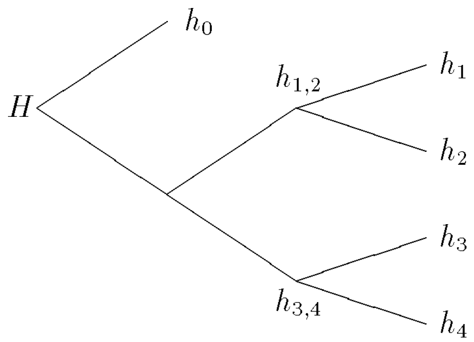
5. Si $c = C$ alors M écrit h_0, h_1, h_3, k, h_3, h_4, c et Answer, sinon M retourne à 1 (reset [7])

Alors M simule un ruban de communication indistingable d'un ruban d'une véritable identification de r tours en un nombre moyen de $3 \times r$ étapes. □

7 Astuces pratiques

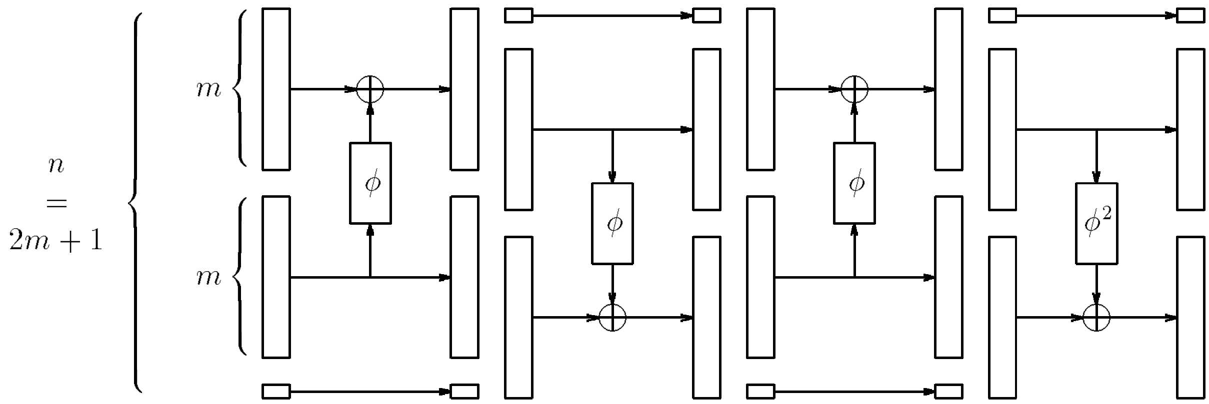
7.1 Les engagements

Deux situations peuvent être envisagées selon que l'on préfère optimiser le temps et la quantité des transferts de données, ou si l'on veut économiser la mémoire vive. Dans le premier cas, on peut effectuer les engagements sous forme d'arbre de hachage –voir schéma ci-dessous–. Le second cas est le cas traité.



7.2 Les permutations

Fabriquer des permutations de façon parfaitement aléatoire n'est pas chose aisée, cependant, on peut utiliser les générateurs de permutations basés sur le schéma du DES, étudiés par Luby et Rackoff [8] ainsi que par Patarin [12]:



$\phi : \mathbb{F}_{2^m} \longrightarrow \mathbb{F}_{2^m}$, aléatoire définit à elle seule la permutation.

7.3 Les objets aléatoires

On suppose que l'on possède un générateur de bits pseudo-aléatoires, public, de la forme $f(s, i)$ où s est le germe, qui suffit à décrire la séquence des bits suivants.
Ce germe sera de longueur 64 bits.
On ne communique, alors, que les germes des générateurs pseudo-aléatoires.

8 Protocoles pratiques

Nous allons nous restreindre aux protocoles à 5 passes.

8.1 Version zeroknowledge

Notation 5

H : *une fonction de hachage résistante aux collisions.*
U : *le vecteur de taille n dont toutes les coordonnées sont égales à 1.*

Prouveur		Vérifieur
P, permutation aléatoire de $\{0,\ldots,m-1\}$ Q, permutation aléatoire signée de $\{0,\ldots,n-1\}$ W, vecteur aléatoire de $\mathbb{F}_p^n$ $A' = PAQ$ $V' = Q^{-1}V$ $h_0 = H(P\|Q)$, $h_1 = H(W\|V')$, $h_2 = H(A'W\|A'V')$, $s_1 = germe(P,Q)$, $s_2 = germe(W)$,	$\xrightarrow{h_0, h_1, h_2}$	
	$\xleftarrow{k}$	$k \in_R \mathbb{F}_p^\star$
$R = kW + V'$, $h_3 = H(R)$, $h_4 = H(A'R)$	$\xrightarrow{h_3, h_4}$	
	$\xleftarrow{c}$	$c \in_R \{0,1,2\}$
Si $c = 0$	$\xrightarrow{s_1, R}$	Vérifie h_0, h_3 et h_4
Si $c = 1$ $T = \frac{1}{2}(A'V' - U)$	$\xrightarrow{A'W, T}$	Vérifie T convenable Vérifie h_2 et h_4 par $h_2 = H(A'W\|2T + U)$ $h_4 = H(kA'W + 2T + U)$
Si $c = 2$	$\xrightarrow{s_2, V'}$	Vérifie V' convenable Vérifie h_1 et h_3 par $h_1 = H(W\|V')$ $h_3 = H(kW + V')$

8.2 Version économique

Une version économique des ces protocoles peut-être faite. Elles visent notamment à restreindre le nombre de tours nécessaires pour atteindre une certaine sécurité. Pour le protocole à 5 passes, on réduit la probabilité de triche de 2 sur 3 à 1 sur 2.

Prouveur		Vérifieur
P, permutation aléatoire de $\{0,\dots,m-1\}$ Q, permutation aléatoire signée de $\{0,\dots,n-1\}$ W, vecteur aléatoire de $\mathbb{F}_p^n$		
$A' = PAQ$ $V' = Q^{-1}V$		
$h_0 = H(P\|Q)$, $h_1 = H(W\|A'W\|V\|A'V')$, $s_1 = germe(P,Q)$, $s_2 = germe(W)$.	$\xrightarrow{h_0,\, h_1}$	
	$\xleftarrow{k}$	$k \in_R \mathbb{F}_p^\star$
$R = kW + V'$, $h_2 = H(R\|A'R)$.	$\xrightarrow{h_2}$	
	$\xleftarrow{c}$	$c \in_R \{0,1\}$
Si $c = 0$	$\xrightarrow{s_1,\, R}$	Vérifie h_0 et $h_2 = H(R\|PAQR)$
Si $c = 1$ $T = \frac{1}{2}(A'V' - U)$	$\xrightarrow{s_2,\, V',\, A'W,\, T}$	Vérifie V' et T convenables Vérifie h_1 et $h_2 = H(X\|Y)$ avec $X = kW + V'$ $Y = kA'W + 2T + U$

Cependant, pour la question $c = 1$, le vérifieur apprend un couple de vecteurs avec leur image par A'. De là, il peut théoriquement déduire quelques bits de A', donc par la même occasion, une information sur P et Q. Comme il connaît V', cela lui «révèle» une fraction de bit de V. Déjà, cette fraction de bit sur V semble difficilement extractible, de plus, P et Q sont différentes à chaque itération, donc cette «information» potentielle semble inutilisable.

Malheureusement, ce protocole ne peut plus être défini «zero-knowledge».

9 Performances

Les performances de ce schéma sont semblables aux performances des schémas linéaires existants :

	SD Stern	CLE Stern	PKP Shamir	**PPP** 3p ZK	**PPP** 5p ZK
Taille de la matrice	256×512	24×24	37×64	101×117	
sur le corps	$\mathbb{F}_2$	$\mathbb{F}_{16}$	$\mathbb{F}_{251}$	$\mathbb{F}_2$	
complexité de la meilleure attaque connue	2^{68}	2^{52}	$> 2^{100}$	2^{64}	
Nombre de tours	35	20	20	48	35
clé publique (bits)	256	80	296	144	
clé secrète (bits)	512	80	384	117	
bits envoyés à chaque tour	954	824	832	896	1040
transmission globale (kilo-octets)	4.08	2.01	2.03	5.25	4.44

- Comme on peut le voir dans ce tableau, avec une clé secrète plus sûre que dans certains autres schémas (le facteur de travail de la meilleure attaque dépasse les 2^{64} opérations élémentaires), et une probabilité de 1 pour 1 million pour un intrus d'être accepté, une identification nécessitera moins de 4.5 kilo-octets de transmissions entre le prouveur et le vérifieur (à comparer à 2 kilo-octets pour PKP et CLE, ainsi que 4 kilo-octets pour SD). Et nous pouvons diminuer cette quantité en utilisant des arbres de hachage.

- De plus, les opérations se résument à des additions et des soustractions entre petits entiers (moins d'un octet), voire modulo 2. Elles sont parfaitement adaptées à l'environnement minimal d'un processeur à 8 bits.

- Si on utilise une matrice, commune à tous, M, stockée dans le germe d'un générateur pseudo-aléatoire, et si les clé sont stockées de la façon suivante :

clé secrète	:	un ε-vecteur aléatoire V de taille n	(moins de 15 octets)
clé publique	:	l'unique ε-vecteur L tel que $L_i(MV)_i \geq 0$ et le multi-ensemble $S = \{\{L_i(MV)_i\}\}$	(18 octets)

  L'espace mémoire nécessaire est vraiment très faible, comparé à PKP ou SD. Mais il ne faut pas perdre de vue que, comme PKP, SD et CLE, ce schéma n'est pas basé sur l'identité. Cela signifie que la clé publique devra être certifiée par une autorité.

- Ces protocoles n'utilisent que des opérations très simples, ainsi le programme ne sera pas très encombrant. De plus, la taille des données (données communes et clés) est très faible. Ainsi, une petit EEPROM sera suffisante.

- Peu de calculs intermédiaires auront à être stockés, donc une RAM minimale suffira.

10 Conclusion

Nous avons défini un nouveau schéma d'identification très simple à implanter sur tout type de carte à puce en raison des opérations élémentaires utilisées et de la faible taille des données. Les attaques sont les bienvenues.

Bibliographie

[1] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and hardness of approximation problems. In *Proceedings of the 33rd Symposium on the Foundations of Computer Science FOCS*, pages 14–23. IEEE, 1992.

[2] M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[3] M. Bellare and M. Sudan. Improved non-approximability results. In *Proceedings of the 26th ACM Symposium on the Theory of Computing STOC*, pages 184–193, 1994.

[4] A. Fiat and A. Shamir. How to prove yourself: practical solutions of identification and signature problems. In *Advances in Cryptology – Proceedings of CRYPTO '86*, volume Lecture Notes in Computer Science 263, pages 186–194. Springer-Verlag, 1987.

[5] M.R. Garey and D.S. Johnson. *Computers and Intractability, A Guide to the Theory of $\mathcal{NP}$-Completeness*. Freeman, New-York, 1979.

[6] O. Goldreich, S. Micali, and A. Wigderson. How to prove all np statements in zero-knowledge and a methodology of cryptographic protocol design. In *Advances in Cryptology – Proceedings of CRYPTO '86*, volume Lecture Notes in Computer Science 263, pages 171–185. Springer-Verlag, 1987.

[7] S. Goldwasser, S. Micali, and C. Rackoff. Knowledge complexity of interactive proof systems. In *Proceedings of the 17th ACM Symposium on the Theory of Computing STOC*, pages 291–304, 1985.

[8] M. Luby and Ch. Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM Journal and Computing*, 17(2):373–386, 1988.

[9] K. Ohta and T. Okamoto. A modification of the fiat-shamir scheme. In *Advances in Cryptology – Proceedings of CRYPTO '88*, volume Lecture Notes in Computer Science 403, pages 232–243. Springer-Verlag, 1989.

[10] H. Ong and C.P. Schnorr. Fast signature generation with a fiat shamir-like scheme. In *Advances in Cryptology – Proceedings of EUROCRYPT '90*, volume Lecture Notes in Computer Science, pages 432–440. Springer-Verlag, 1991.

[11] C. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and Systems Sciences*, 43:425–440, 1991.

[12] J. Patarin. *Etude des g n rateurs de permutations pseudo-al atoires bas s sur le sch ma du DES*. PhD thesis, Universit de Paris VII, november 1991.

[13] J.J. Quisquater and L. Guillou. A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory. In *Advances in Cryptology – Proceedings of EUROCRYPT '88*, volume Lecture Notes in Computer Science 330, pages 123–128. Springer-Verlag, 1989.

[14] C.P. Schnorr. Efficient identification and signatures for smart cards. In *Advances in Cryptology – Proceedings of CRYPTO '89*, volume Lecture Notes in Computer Science 435, pages 235–251. Springer-Verlag, 1990.

[15] A. Shamir. An efficient identification scheme based on permuted kernels. In *Advances in Cryptology – Proceedings of CRYPTO '89*, volume Lecture Notes in Computer Science 435, pages 606–609. Springer-Verlag, 1990.

[16] M. Skubiszewski. *Optimisation par recuit simul : mise en œuvre mat rielle de la machine de Boltzmann, application l' tude des suites synchronisantes.* PhD thesis, Universit d'Orsay, june 1993.

[17] J. Stern. Designing identification schemes with keys of short size. In *Advances in Cryptology – proceedings of CRYPTO '94*, volume Lecture Notes in Computer Science 839, pages 164–173. Springer-Verlag, 1994.

[18] J. Stern. A new identification scheme based on syndrome decoding. In *Advances in Cryptology – proceedings of CRYPTO '93*, volume Lecture Notes in Computer Science 773, pages 13–21. Springer-Verlag, 1994.