MINES Saint-Étienne
Une école de l'IMT
INSPIRING INNOVATION SINCE 1816

WISeKey

Comment construire une attaque SPA contre le pivot de Gauss

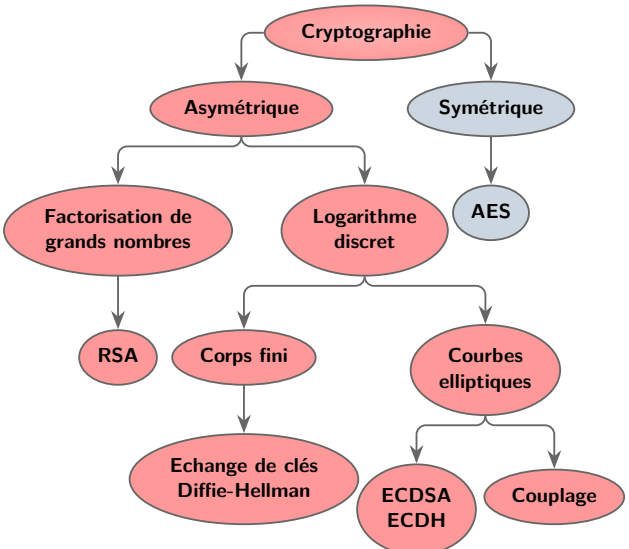
Nadia El Mrabet, SAS EMSE
<nadia.el-mrabet@emse.fr>

10 november 2021, Paris

Travail en collaboration avec : Lina Mortajine (EMSE - Wisekey)
Pierre-Louis Cayrel (LHC, Univ. Jean-Monnet)
Agathe Cherrière (Irisa)
Jérôme Lablanche (Wisekey)
Tania Richmond (DGA)

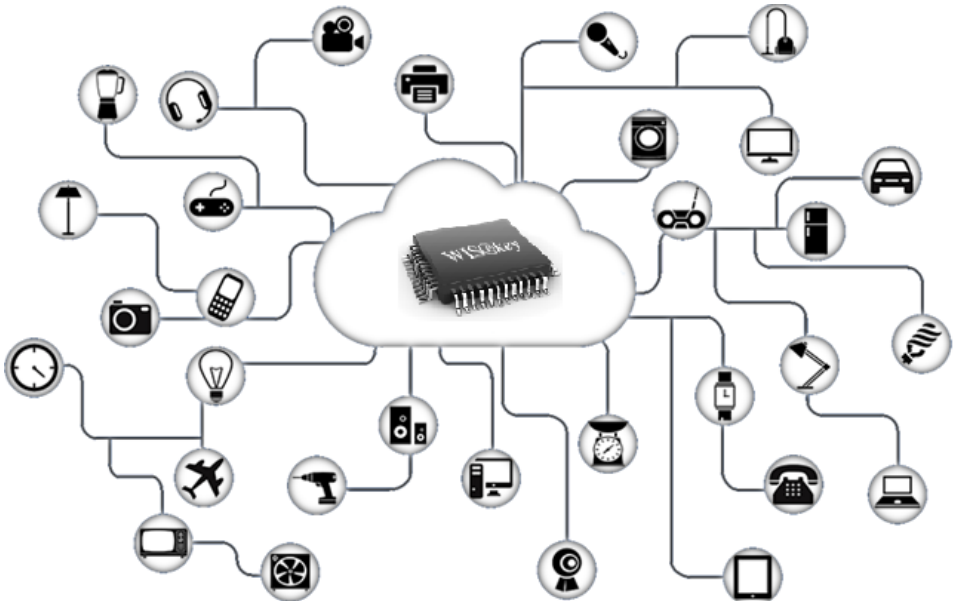
Sommaire

Cryptographie asymétrique - aujourd'hui



- Échange de clés
- Chiffrement à clé publique
- Signature

Cryptographie asymétrique - aujourd'hui



Cryptographie asymétrique - demain

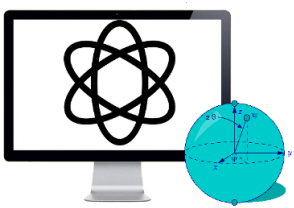
Menace pour la cryptographie asymétique

Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer*

Peter W. Shor†

Abstract

A digital computer is generally believed to be an efficient universal computing device; that is, it is believed able to simulate any physical computing device with an increase in computation time by at most a polynomial factor. This may not be true when quantum mechanics is taken into consideration. This paper considers factoring integers and finding discrete logarithms, two problems which are generally thought to be hard on a classical computer and which have been used as the basis of several proposed cryptosystems. Efficient randomized algorithms are given for these two problems on a hypothetical quantum computer. These algorithms take a number of steps polynomial in the input size, e.g., the number of digits of the integer to be factored.



Cryptographie asymétrique - demain

Processeur quantique de 133 qbits © IBM

Cryptographie asymétrique - demain

Les familles post-quantiques

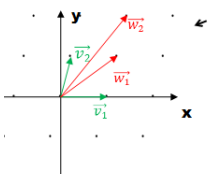
Polynômes multivariés

$$\begin{cases} x_1^2 + x_1x_2 + x_3^2 + x_3x_4 + x_4 = 1 \\ x_1x_2 + x_2^2 + x_2x_3 + x_2x_4 + x_3^2 + x_4^2 = 0 \\ x_1x_4 + x_2^2 + x_2x_3 + x_2x_4 + x_3^2 + x_4^2 = 1 \\ x_1^2 + x_1x_3 + x_2x_3 + x_4^2 = 1 \end{cases}$$

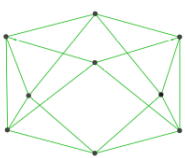
Codes correcteurs

Cryptographie post-quantique

Réseaux euclidiens



Graphes d'isogénies



Cryptographie asymétrique - demain

Quelques incertitudes

- ◇ Manque de preuves de sécurité pour certains schémas (cryptographie récente)
- ◇ Peu d'implantations embarquées
- ◇ Taille des données manipulées beaucoup plus importantes que celles utilisées dans les cryptosystèmes actuellement déployés
- ◇ Peu études d'attaques physiques sur les implémentations

Standardisation PQC du NIST

Objectif de la standardisation

Standardiser de nouveaux cryptosystèmes résistants aux attaques classiques et quantiques existantes

- 2016 – Annonce de la standardisation
- 2017 – Début du premier tour (69 candidats)
- 2018 – Première conférence PQC
- 2019 – Début du second tour (26 candidats)
  Deuxième conférence PQC
- 2020 – Début du troisième tour (7 finalistes)

NIST
National Institute of Standards and Technology
U.S. Department of Commerce

- 2021 – Troisième conférence PQC
- 2022 – Choix parmi les finalistes
- 2024 – Premières versions des standards

Sommaire

Étude du candidat ROLLO

ROLLO / soumission fin du second tour [AMAB+20] :

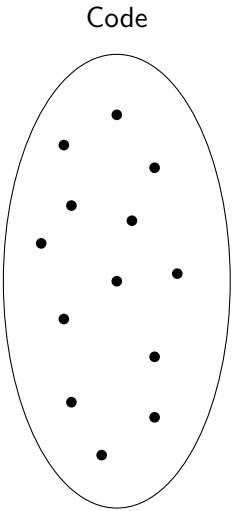
- ◇ 2 schémas : un KEM (ROLLO-I) et un PKE (ROLLO-II)
- ◇ nouveaux paramètres

Schéma	clé publique	clé privée	chiffré
ROLLO-I-128	696	1392	696
ROLLO-I-192	958	1916	958
ROLLO-I-256	1371	2742	1371

TABLE – Taille des paramètres (en octets) - nouvelle version

- ◇ Non finaliste de la standardisation mais toujours intéressant
- ◇ Implantation en temps constant fournie dans la nouvelle version

Codes correcteurs d'erreurs



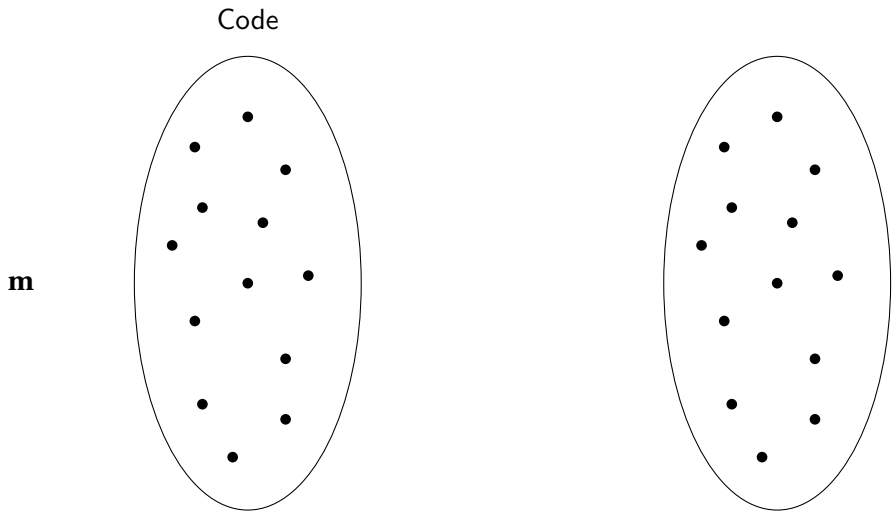
Code

Soit $k, n \in \mathbb{N}$.

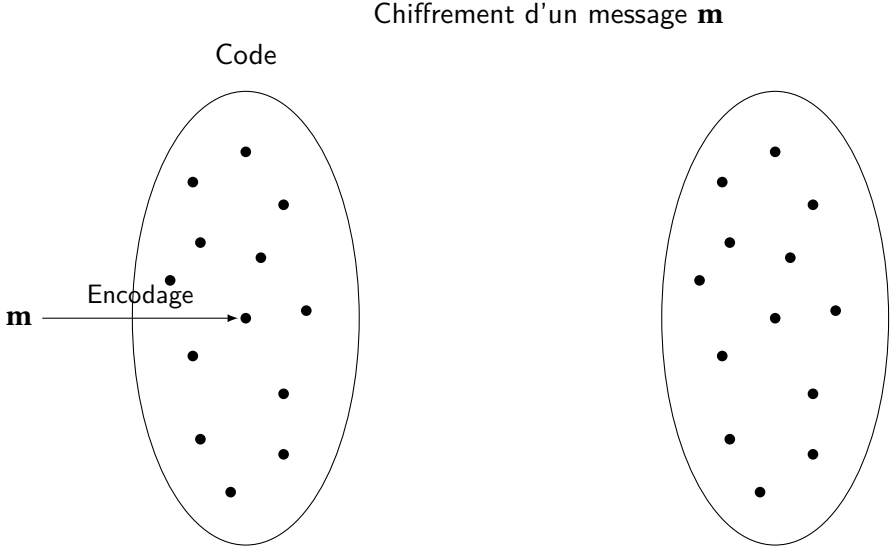
- Un code linéaire $\mathscr{C}$ sur $\mathbb{F}_q$ est un sous-espace vectoriel de dimension k de $\mathbb{F}_q^n$
- Un élément de l'ensemble $\mathscr{C}$ est appelé *mot de code*

Codes correcteurs d'erreurs

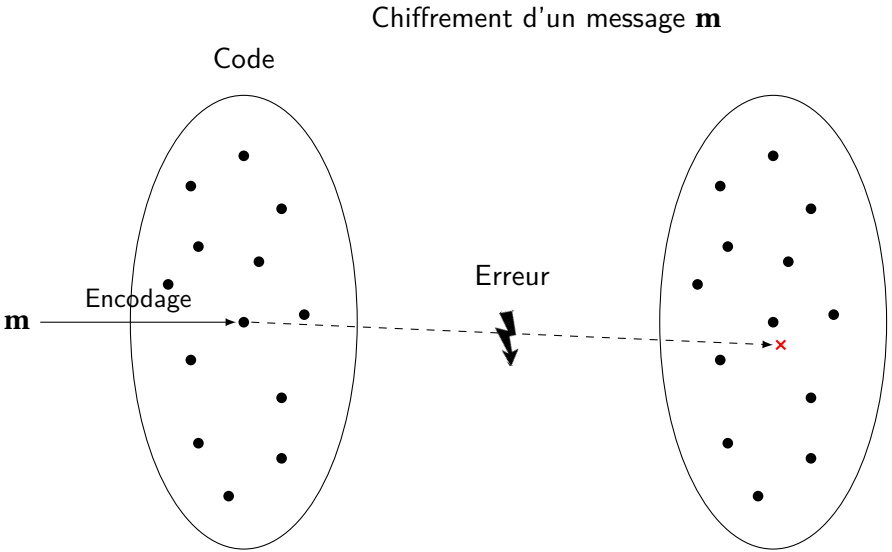
Chiffrement d'un message $\mathbf{m}$



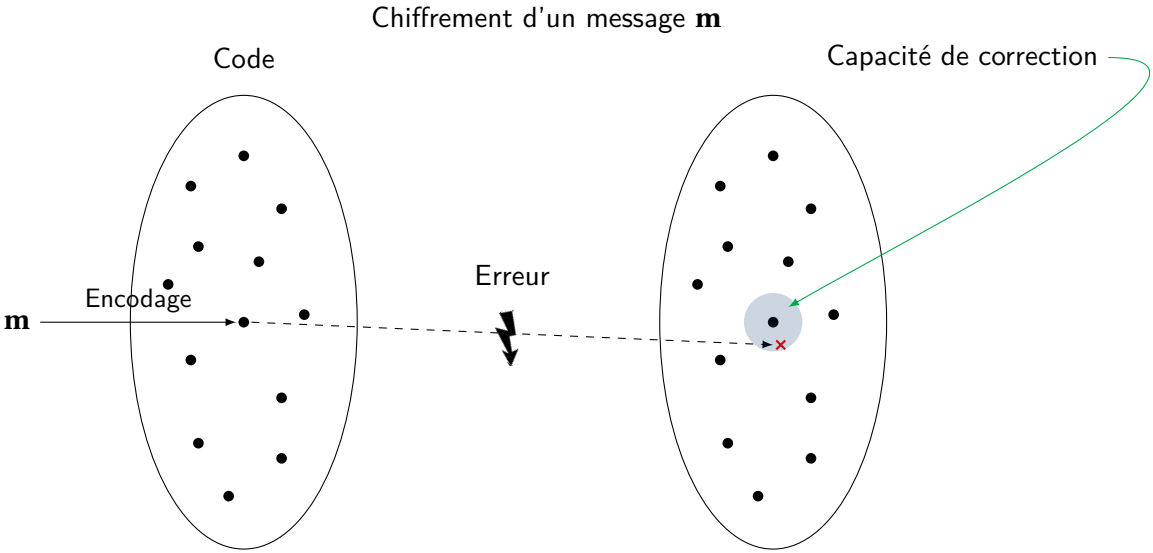
Codes correcteurs d'erreurs



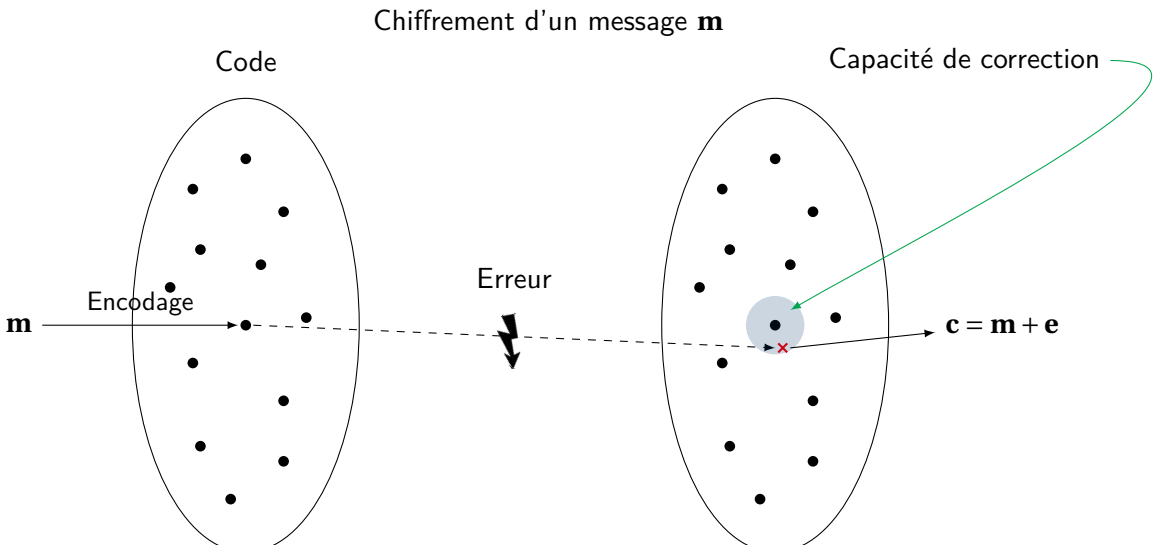
Codes correcteurs d'erreurs



Codes correcteurs d'erreurs



Codes correcteurs d'erreurs



Codes correcteurs d'erreurs

Distance et capacité de correction d'un code $\mathscr{C}$

- $d_{\min} = \min_{\substack{(\mathbf{x},\mathbf{y})\in\mathscr{C}^2 \\ \mathbf{x}\neq\mathbf{y}}} d(\mathbf{x},\mathbf{y})$
- $t = \left\lfloor \frac{d_{\min}-1}{2} \right\rfloor$

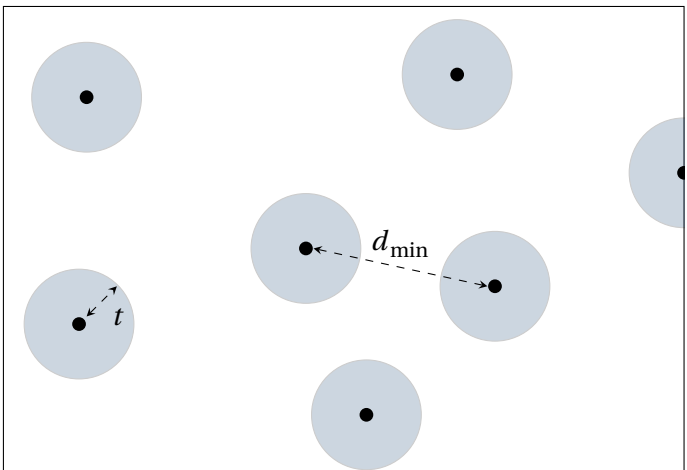
Codes correcteurs d'erreurs

Distance et capacité de correction d'un code $\mathscr{C}$

- $d_{\min} = \min_{\substack{(\mathbf{x},\mathbf{y})\in\mathscr{C}^2 \\ \mathbf{x}\neq\mathbf{y}}} d(\mathbf{x},\mathbf{y})$
- $t = \left\lfloor \frac{d_{\min}-1}{2} \right\rfloor$

Métriques

- Hamming [Ham50]
- Rang [Del78]



Codes correcteurs en métrique rang

Soit $\mathscr{C}$ un code de longueur n tel que ses coefficients soient dans $\mathbf{F}_{q^m}$.

Représentation d'un mot de code

$$\mathbf{x} = (x_0, \cdots, x_{n-1}) \in \mathscr{C} \quad \longrightarrow \quad M_{\mathbf{x}} = \begin{pmatrix} x_{0,0} & \cdots & x_{0,m-1} \\ \vdots & & \vdots \\ x_{n-1,0} & \cdots & x_{n-1,m-1} \end{pmatrix}$$

Codes correcteurs en métrique rang

Soit $\mathscr{C}$ un code de longueur n tel que ses coefficients soient dans $\mathbf{F}_{q^m}$.

Représentation d'un mot de code

$$\mathbf{x} = (x_0, \cdots, x_{n-1}) \in \mathscr{C} \quad \longrightarrow \quad M_{\mathbf{x}} = \begin{pmatrix} x_{0,0} & \cdots & x_{0,m-1} \\ \vdots & & \vdots \\ x_{n-1,0} & \cdots & x_{n-1,m-1} \end{pmatrix}$$

En théorie

- $\mathsf{Rang}(\mathbf{x}) = \mathsf{Rang}(M_{\mathbf{x}})$
- $\mathsf{Supp}(\mathbf{x}) = \langle x_0, \cdots, x_{n-1} \rangle_{\mathbb{F}_q}$

Codes correcteurs en métrique rang

Soit $\mathscr{C}$ un code de longueur n tel que ses coefficients soient dans $\mathbf{F}_{q^m}$.

Représentation d'un mot de code

$$\mathbf{x} = (x_0, \cdots, x_{n-1}) \in \mathscr{C} \quad \longrightarrow \quad M_{\mathbf{x}} = \begin{pmatrix} x_{0,0} & \cdots & x_{0,m-1} \\ \vdots & & \vdots \\ x_{n-1,0} & \cdots & x_{n-1,m-1} \end{pmatrix}$$

En théorie

- $\mathrm{Rang}(\mathbf{x}) = \mathrm{Rang}(M_{\mathbf{x}})$
- $\mathrm{Supp}(\mathbf{x}) = \langle x_0, \cdots, x_{n-1} \rangle_{\mathbb{F}_q}$

En pratique

Pivot de Gauss

Codes correcteurs en métrique rang

Méthode du pivot de Gauss

Opérations utilisées :

- ajout d'une ligne à une autre
- multiplication d'une ligne par un scalaire non nul
- échange de deux lignes

Matrice après réduction

$$\begin{pmatrix} 1 & \star & \star & \star & \star \\ 0 & 1 & \star & \star & \star \\ \vdots & \vdots & \ddots & \star & \star \\ 0 & 0 & \cdots & 1 & \star \end{pmatrix}$$

Codes correcteurs en métrique rang

Méthode du pivot de Gauss

Opérations utilisées :

- ajout d'une ligne à une autre
- multiplication d'une ligne par un scalaire non nul
- échange de deux lignes

Exemple sur une matrice binaire

$$M_{\mathbf{x}} = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix}$$

Matrice après réduction

$$\begin{pmatrix} 1 & \star & \star & \star & \star \\ 0 & 1 & \star & \star & \star \\ \vdots & \vdots & \ddots & \star & \star \\ 0 & 0 & \cdots & 1 & \star \end{pmatrix}$$

Codes correcteurs en métrique rang

Méthode du pivot de Gauss

Opérations utilisées :

- ajout d'une ligne à une autre
- multiplication d'une ligne par un scalaire non nul
- échange de deux lignes

Exemple sur une matrice binaire

$$M_{\mathbf{x}} = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix} \xrightarrow{\text{réduction}} \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Matrice après réduction

$$\begin{pmatrix} 1 & \star & \star & \star & \star \\ 0 & 1 & \star & \star & \star \\ \vdots & \vdots & \ddots & \star & \star \\ 0 & 0 & \cdots & 1 & \star \end{pmatrix}$$

Codes correcteurs en métrique rang

Méthode du pivot de Gauss

Opérations utilisées :

- ajout d'une ligne à une autre
- multiplication d'une ligne par un scalaire non nul
- échange de deux lignes

Matrice après réduction

$$\begin{pmatrix} 1 & \star & \star & \star & \star \\ 0 & 1 & \star & \star & \star \\ \vdots & \vdots & \ddots & \star & \star \\ 0 & 0 & \cdots & 1 & \star \end{pmatrix}$$

Exemple sur une matrice binaire

$$M_{\mathbf{x}} = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix} \xrightarrow{\text{réduction}} \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \longrightarrow \begin{cases} \text{Rang}(\mathbf{x}) = 3 \\ \text{Supp}(\mathbf{x}) = \\ \langle (1,0,1,1), (1,1,0,1), (1,0,0,1) \rangle_{\mathbb{F}_q} \end{cases}$$

Sommaire

ROLLO-I scheme

Alice *Bob*

Génération des clés

Génération du support $F \in \mathbb{F}_{q^m}^n$ de rang d.

Génération de la clé privée :

$\mathbf{sk} = (\mathbf{x}, \mathbf{y})$ à partir de F.

Calcul de la clé publique

$\mathbf{h} = \mathbf{x}^{-1} \cdot \mathbf{y} \mod P_n$.

ROLLO-I scheme

Alice

Génération des clés

Génération du support $F \in \mathbb{F}_{q^m}^n$ de rang d.

Génération de la clé privée :

$\mathbf{sk} = (\mathbf{x}, \mathbf{y})$ à partir de F.

Calcul de la clé publique

$\mathbf{h} = \mathbf{x}^{-1} \cdot \mathbf{y} \mod P_n$.

$\xrightarrow{\mathbf{h}}$

Bob

Encapsulation

Génération d'un support E de rang r,

de deux erreurs$(\mathbf{e_1}, \mathbf{e_2})$ à partir de E.

Calcul de $\mathbf{c} = \mathbf{e_2} + \mathbf{e_1} \cdot \mathbf{h} \mod P_n$

Dérivation du secret partagé :

$K = \mathsf{Hash}(E)$

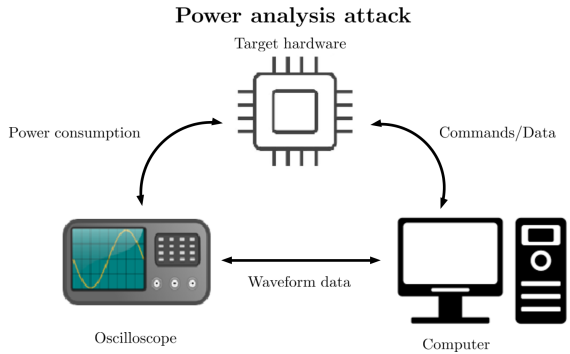
ROLLO-I scheme

Alice		*Bob*
		Encapsulation
		Génération d'un support E de rang r, de deux erreurs $(\mathbf{e_1}, \mathbf{e_2})$ à partir de E
		Calcul de $\mathbf{c} = \mathbf{e_2} + \mathbf{e_1} \cdot \mathbf{h} \mod P_n$.
		$K = \mathsf{Hash}(E)$.
	$\xleftarrow{\mathbf{c}}$	
Decapsulation		
Calculer le syndrome :		
$\mathbf{s} = \mathbf{x} \cdot \mathbf{c} \mod P_n$.		
Retrouver le support de l'erreur :		
$E = \mathsf{RSR}(F, \mathbf{s}, r)$.		
Dériver le secret partagé :		
$K = \mathsf{Hash}(E)$.		

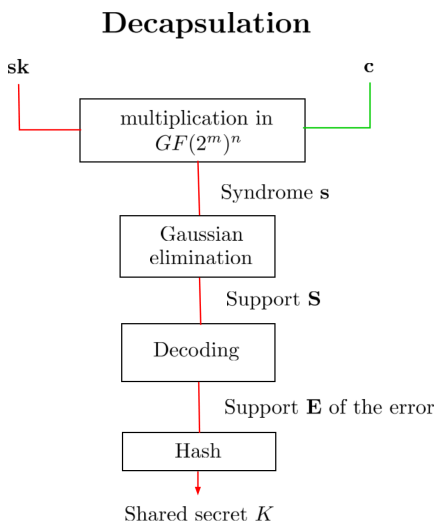
Sommaire

Cible de l'attaque

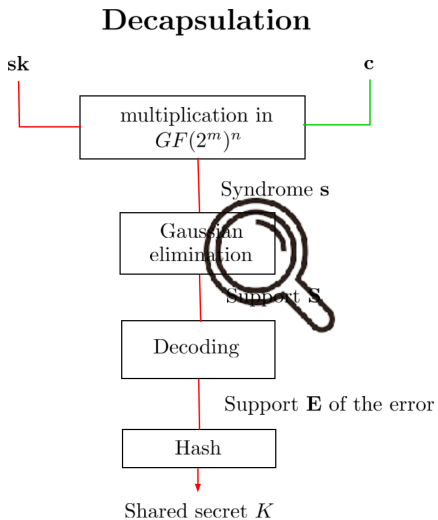
Passive attacks



Cible de l'attaque



Cible de l'attaque



Entrées : Matrix $M \in \mathcal{M}_{n,m}(\mathbb{F}_2)$
Output : Matrice échelonnée

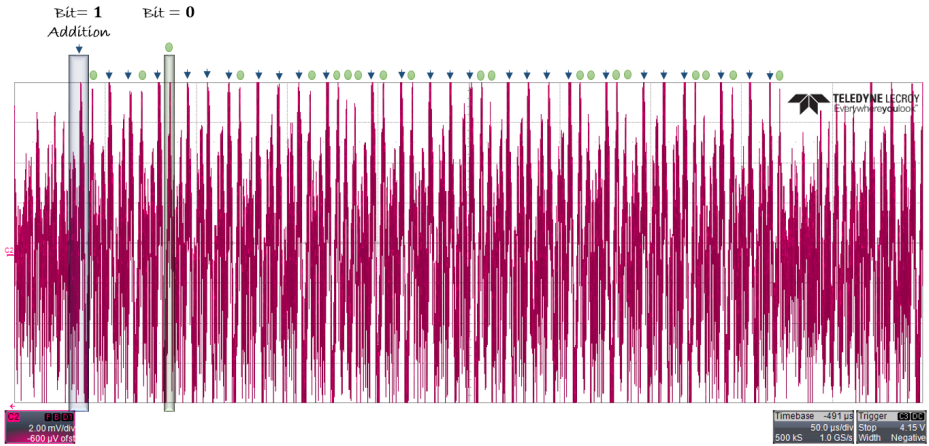
```
1  Dim ← 0
2  pour i = 1 to m faire
3      pour j = 1 to n faire
4          si M_{j,i} = 1 alors
5              line i ↔ line j ;                    // La ligne j est un pivot.
6              Dim ← Dim +1
7              break
8      pour k = ligne i + 1 to n faire
9          si M_{k,i} = 1 alors
10             ligne k ← ligne k + ligne i
11 retourner (M, Dim)
```

Algorithme 1 : Élimination Gaussianne

Attaque par canaux auxiliaires contre ROLLO-I-128

Exemple de la reconstruction de la matrice syndrome á

$$10110101110100010101110011111001001110010110.$$



Attaque par canaux auxiliaires contre ROLLO-I-128

Suite á l'élimination Gaussienne, nous obtenons la matrice

$$M_{s,0} = \begin{pmatrix} s_{0,0} & * & * & * & * & * \\ s_{1,0} & s_{1,1} & * & * & * & * \\ \vdots & \vdots & \ddots & * & * & * \\ s_{46,0} & s_{46,1} & \cdots & s_{46,29} & * & * \end{pmatrix}$$

Attaque par canaux auxiliaires contre ROLLO-I-128

Comment récupérer les autres coefficients ?

Attaque par canaux auxiliaires contre ROLLO-I-128

Comment récupérer les autres coefficients ?

Rotations de la matrice $M_{s,0}$

Attaque par canaux auxiliaires contre ROLLO-I-128

Comment récupérer les autres coefficients ?

Rotations de la matrice $M_{s,0}$

→ Multiplication du chiffré par x^i in $\in \mathbb{F}_{2^m}[X]/(P_n)$

$$\begin{aligned} \mathbf{c}_1 &= x.\mathbf{c} \\ \mathbf{c}_2 &= x^2.\mathbf{c} \\ &\vdots \\ \mathbf{c}_{78} &= x^{78}.\mathbf{c} \end{aligned}$$

Attaque par canaux auxiliaires contre ROLLO-I-128

Comment récupérer les autres coefficients ?

Rotations de la matrice $M_{s,0}$

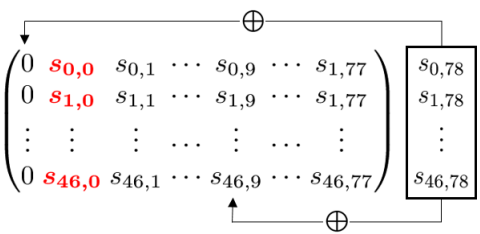
→ Multiplication du chiffré par x^i in $\in \mathbb{F}_{2^m}[X]/(P_n)$

$$\begin{aligned} \mathbf{c}_1 &= x.\mathbf{c} \\ \mathbf{c}_2 &= x^2.\mathbf{c} \\ &\vdots \\ \mathbf{c}_{78} &= x^{78}.\mathbf{c} \end{aligned}$$

→ Implique m différentes matrices $M_{\mathbf{s},i}$ avec $0 \leq i < 79$

Attaque par canaux auxiliaires contre ROLLO-I-128

Rappelons le modulo $P_m(x) = x^{79} + x^9 + 1$,
considérant la matrice $M_{s,1}$, nous obtenons



Faire attention, la colonne 0 correspond au résultat du Xor entre la colonne 9 (cherchée) et la colonne 78 (trouvée).

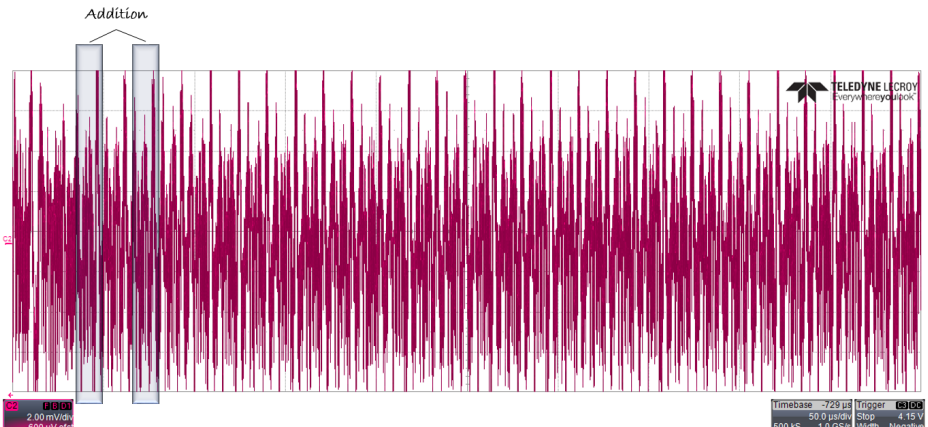
Contremesures

→ Ajouter du bruit afin de rendre les fuites indépendantes des donnés manipulées.

Deux méthodes :

- Rendre aléatoire la recherche du pivot dans la colonne et le traitement des lignes.
- Ajouter des opérations factices si le digit courant est á 0.

Application de la contremesure



Conséquences des contremesures

		Decap	
Niveau de sécurité		Avec contremesures	Sans contremesures
ROLLO-I-128	cycles ($\times 10^6$)	6.43	4.31
	ms	128.6	86.3
ROLLO-I-192	cycles ($\times 10^6$)	12.54	7.8
	ms	250.8	156
ROLLO-I-256	cycles ($\times 10^6$)	23.92	15.54
	ms	478.4	310.8

Le temps d'exécution augmente d'environ $\rightarrow 30\%$.

Sommaire

Sécurité de ROLLO

Trois implantations du pivot de Gauss :

- ◇ Implantation du pivot de Gauss standard
- ◇ Implantation en temps constant [1] utilisant la bibliothèque *rbc* : https://rbc-lib.org/index.html
- ◇ Implantation en temps constant publiée sur *GitHub* (ROLLO-I-128) [2]

$$\begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

1. répertoire rollo-submission_2020-04-21/Additionnal_Implementations
2. https://github.com/peacker/constant_time_rollo

Sécurité de ROLLO

Trois implantations du pivot de Gauss :

- Implantation du pivot de Gauss standard
  → attaque avec m traces et chiffrés choisis
- Implantation en temps constant [1] utilisant la bibliothèque *rbc* : https://rbc-lib.org/index.html
- Implantation en temps constant publiée sur *GitHub* (ROLLO-I-128) [2]

$$\begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix}$$

1. répertoire rollo-submission_2020-04-21/Additionnal_Implementations
2. https://github.com/peacker/constant_time_rollo

Sécurité de ROLLO

Trois implantations du pivot de Gauss :

- ◇ Implantation du pivot de Gauss standard
  → attaque avec m traces et chiffrés choisis
- ◇ Implantation en temps constant [1] utilisant la bibliothèque *rbc* : `https://rbc-lib.org/index.html`
- ◇ Implantation en temps constant publiée sur *GitHub* (ROLLO-I-128) [2]
  → attaque avec 1 trace

$$\begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix}$$

1. répertoire rollo-submission_2020-04-21/Additionnal_Implementations
2. `https://github.com/peacker/constant_time_rollo`

Implantation du pivot de Gauss

→ Support du syndrome $\mathbf{s} \in \mathbb{F}_{2^m}^n$ représenté par une matrice binaire de taille $n \times m$

$$M_{\mathbf{s}} = \begin{pmatrix} s_{0,0} & \cdots & s_{0,m-1} \\ \vdots & & \vdots \\ s_{n-1,0} & \cdots & s_{n-1,m-1} \end{pmatrix}$$

Notations

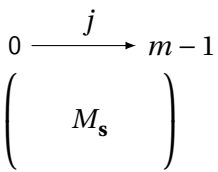
$s_{i,j}$: l'élément à la i^{e} ligne et j^{e} colonne de la matrice $M_{\mathbf{s}}$
s_i : la i^{e} ligne de la matrice $M_{\mathbf{s}}$
$M_{\mathbf{s}}[j]$: la j^{e} colonne de la matrice $M_{\mathbf{s}}$
$M_{\mathbf{s},j}$: matrice obtenue après la j^{e} itération
$\oplus$: XOR entre deux coefficients ou deux lignes de la matrice $M_{\mathbf{s}}$
$\otimes$: multiplication d'une ligne de la matrice $M_{\mathbf{s}}$ par un scalaire

Sécurité de ROLLO

Implantation de référence en temps constant (première boucle interne)

$dimension$: indice de la ligne pivot
$dimension = 0$
Pour $j = 0, \dots, m-1$

$pivot = min(dimension, n-1)$



Sécurité de ROLLO

Implantation de référence en temps constant (première boucle interne)

$dimension$: indice de la ligne pivot
$dimension = 0$
Pour $j = 0, \dots, m-1$

$pivot = min(dimension, n-1)$

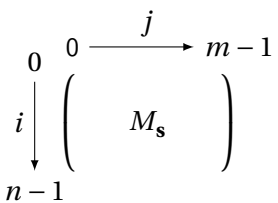
Pour $i = 0, \dots, n-1$

$mask = s_{pivot,j} \oplus s_{i,j}$
$tmp = mask \otimes s_i$
Si $i > pivot$

$s_{pivot} = s_{pivot} \oplus tmp$

Sinon

$dummy = s_{pivot} \oplus tmp$



Sécurité de ROLLO

Implantation de référence en temps constant (première boucle interne)

$dimension$: indice de la ligne pivot
$dimension = 0$
Pour $j = 0, \dots, m-1$

$pivot = min(dimension, n-1)$

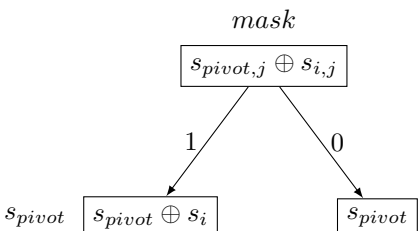
Pour $i = 0, \dots, n-1$

$mask = s_{pivot,j} \oplus s_{i,j}$
$tmp = mask \otimes s_i$
Si $i > pivot$

$s_{pivot} = s_{pivot} \oplus tmp$

Sinon

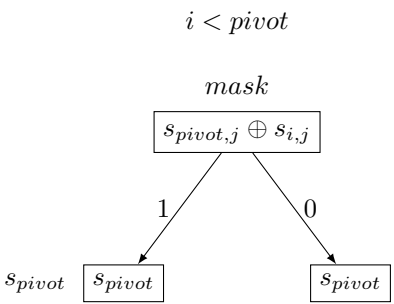
$dummy = s_{pivot} \oplus tmp$



Sécurité de ROLLO

Implantation de référence en temps constant (première boucle interne)

$dimension$: indice de la ligne pivot
$dimension = 0$
Pour $j = 0, \dots, m-1$
    $pivot = min(dimension, n-1)$
    Pour $i = 0, \dots, n-1$
        $mask = s_{pivot,j} \oplus s_{i,j}$
        $tmp = mask \otimes s_i$
        Si $i > pivot$
            $s_{pivot} = s_{pivot} \oplus tmp$
        Sinon
            $dummy = s_{pivot} \oplus tmp$



Sécurité de ROLLO

Implantation de référence en temps constant (première boucle interne)

$dimension$: indice de la ligne pivot
$dimension = 0$
Pour $j = 0, \dots, m-1$
    $pivot = min(dimension, n-1)$
    Pour $i = 0, \dots, n-1$
        $mask = s_{pivot,j} \oplus s_{i,j}$
        $tmp = mask \otimes s_i$
        Si $i > pivot$
            $s_{pivot} = s_{pivot} \oplus tmp$
        Sinon
            $dummy = s_{pivot} \oplus tmp$

Première boucle interne Pour

Les valeurs de $mask \rightarrow$ quelles lignes sont additionnées à la ligne pivot

Sécurité de ROLLO

Implantation de référence en temps constant (deuxième boucle interne)

Pour $j = 0, \dots, m-1$

$pivot = min(dimension, n-1)$

Pour $i = 0, \dots, n-1$

Si $i \neq j$

$mask = s_{i,j}$

$tmp = mask \otimes s_{pivot}$

Si $dimension < n$

$s_i = s_i \oplus tmp$

Sinon

$dummy = s_i \oplus tmp$

$dimension = dimension + s_{pivot,j}$

Sécurité de ROLLO

Implantation de référence en temps constant (deuxième boucle interne)

Pour $j = 0, \dots, m-1$

$pivot = min(dimension, n-1)$

Pour $i = 0, \dots, n-1$

Si $i \neq j$

$mask = s_{i,j}$

$tmp = mask \otimes s_{pivot}$

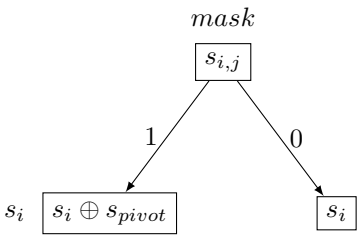
Si $dimension < n$

$s_i = s_i \oplus tmp$

Sinon

$dummy = s_i \oplus tmp$

$dimension = dimension + s_{pivot,j}$



Sécurité de ROLLO

Implantation de référence en temps constant (deuxième boucle interne)

Pour $j = 0, \dots, m-1$

$pivot = min(dimension, n-1)$

Pour $i = 0, \dots, n-1$

Si $i \neq j$

$mask = s_{i,j}$

$tmp = mask \otimes s_{pivot}$

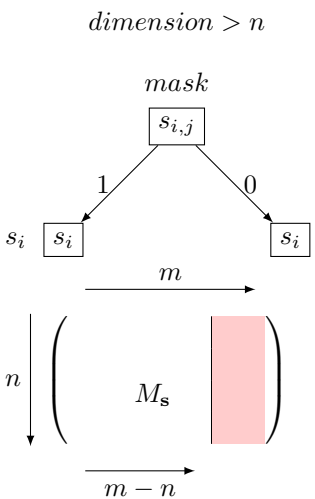
Si $dimension < n$

$s_i = s_i \oplus tmp$

Sinon

$dummy = s_i \oplus tmp$

$dimension = dimension + s_{pivot,j}$



Sécurité de ROLLO

Implantation de référence en temps constant (deuxième boucle interne)

Pour $j = 0, \dots, m-1$

$pivot = min(dimension, n-1)$

Pour $i = 0, \dots, n-1$

Si $i \neq j$

$mask = s_{i,j}$

$tmp = mask \otimes s_{pivot}$

Si $dimension < n$

$s_i = s_i \oplus tmp$

Sinon

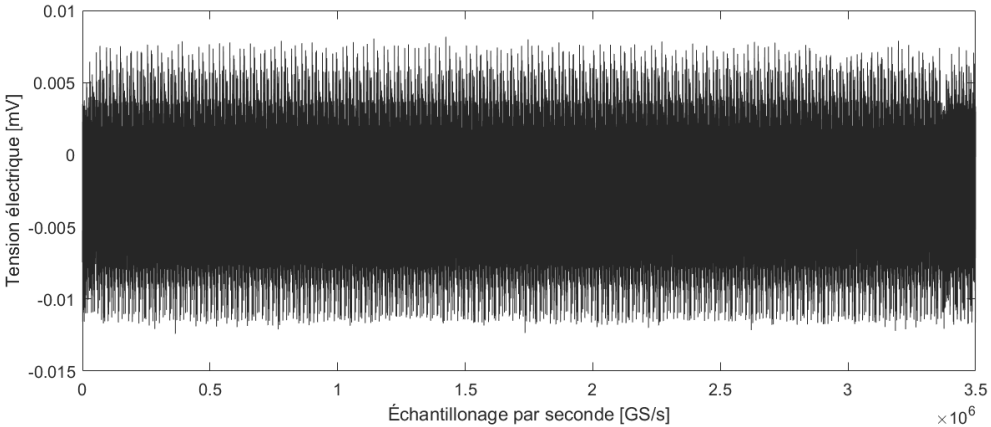
$dummy = s_i \oplus tmp$

$dimension = dimension + s_{pivot,j}$

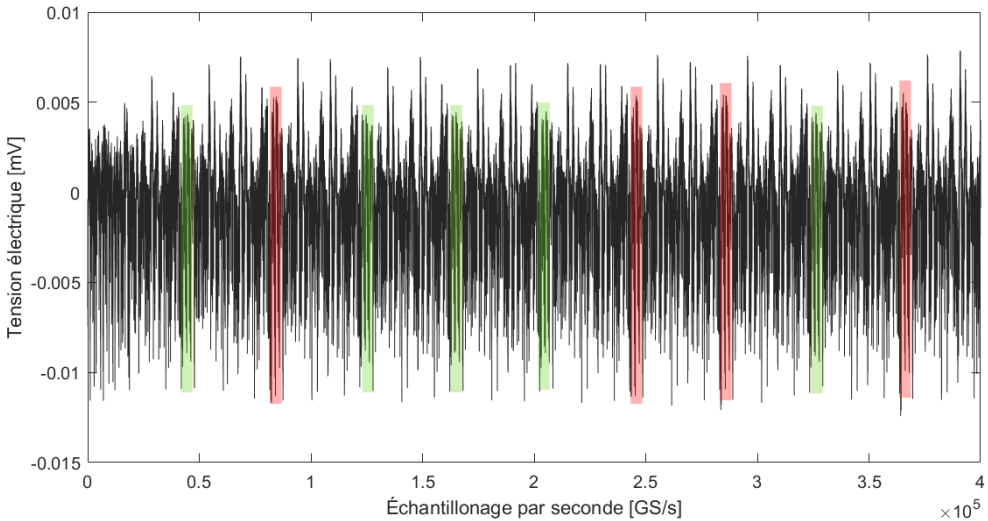
Seconde boucle interne Pour

Les valeurs de $mask \rightarrow$ quelles lignes sont réduites

Trace de la première boucle sur CORTEX-M3



Trace de la première boucle sur CORTEX-M3



Sécurité de ROLLO

→ Comment retrouver la matrice syndrome initiale à partir des valeurs des masques ?

valeurs de $mask$ après le traitement de la j^{e} colonne

Première boucle : $\delta_j = (\delta_{0,j}, \dots, \delta_{j-1,j}, *, \delta_{j+1,j}, \dots, \delta_{n-1,j})$
Deuxième boucle : $\delta'_j = (\delta'_{0,j}, \dots, \delta'_{j-1,j}, *, \delta'_{j+1,j}, \dots, \delta'_{n-1,j})$
où $*$ représente le pivot

$$M_{\mathbf{s}} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Valeurs de $mask$ - première boucle :
$(*,1,1,1,0,0,0), (1,*,1,0,1,1,0), (1,0,*,0,1,0,1), (1,1,1,*,0,1,1),$
$(1,1,1,0,*,1,0)$
Valeurs de $mask$ - deuxième boucle :
$(*,0,0,0,1,1,1), (1,*,1,1,0,0,1), (0,1,*,1,0,1,0), (1,1,1,*,0,1,0),$
$(1,1,1,0,*,1,1)$

Sécurité de ROLLO

Valeurs de $mask$ de la première boucle interne **Pour**

- $\delta_0 = (1,1,1,1,0,0,0) \implies s_0 \leftarrow s_0 + s_1 + s_2 + s_3$

En définissant

$$J_k = \begin{pmatrix} 1 & 0 & \dots & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 & \dots & 0 \\ \vdots & \vdots & & \vdots & & \vdots \\ \delta_{0,k} & \delta_{1,k} & \dots & 1 & \dots & \delta_{n-1,k} \\ \vdots & \vdots & & \vdots & & \vdots \\ 0 & 0 & \dots & 0 & \dots & 1 \end{pmatrix}$$

À la suite de l'exécution de la première boucle, nous obtenons la matrice

$$M'_{\mathbf{s}} = J_0 \times M_{\mathbf{s}} = \left(\begin{array}{c|cccccc} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ \hline \mathbf{0} & \multicolumn{6}{c}{I_6} \end{array}\right) \times M_{\mathbf{s}}$$

Sécurité de ROLLO

Valeurs de $mask$ de la deuxième boucle interne **Pour**

$\diamond$ $\delta'_0 = (1,0,0,0,1,1,1) \Longrightarrow \begin{cases} s'_{0,0} = 1 \\ s'_{1,0} = 0 \\ s'_{2,0} = 0 \\ s'_{3,0} = 0 \\ s'_{4,0} = 1 \\ s'_{5,0} = 1 \\ s'_{6,0} = 1 \end{cases}$

Ce qui revient à résoudre le système d'équations [1] :

$$M'_{\mathbf{s}}[0] = (1\ \ 0\ \ 0\ \ 0\ \ 1\ \ 1\ \ 1)^T$$

1. Utilisation du solveur linéaire de *SageMath*.

Sécurité de ROLLO

Valeurs de $mask$ de la deuxième boucle interne **Pour**

- $\delta'_0 = (1,0,0,0,1,1,1) \Longrightarrow \begin{cases} s'_{0,0} = 1 \\ s'_{1,0} = 0 \\ s'_{2,0} = 0 \\ s'_{3,0} = 0 \\ s'_{4,0} = 1 \\ s'_{5,0} = 1 \\ s'_{6,0} = 1 \end{cases}$

  Ce qui revient à résoudre le système d'équations [1] :

$$M'_{\mathbf{s}}[0] = (1 \;\; 0 \;\; 0 \;\; 0 \;\; 1 \;\; 1 \;\; 1)^T$$

- $\delta'_0 = (1,0,0,0,1,1,1) \Longrightarrow \begin{cases} s'_4 \leftarrow s'_4 + s'_0 \\ s'_5 \leftarrow s'_5 + s'_0 \\ s'_6 \leftarrow s'_6 + s'_0 \end{cases}$

1. Utilisation du solveur linéaire de *SageMath*.

Sécurité de ROLLO

En définissant

$$J'_k = \begin{pmatrix} 1 & 0 & \dots & \delta'_{0,k} & \dots & 0 \\ 0 & 1 & \dots & \delta'_{1,k} & \dots & 0 \\ \vdots & \vdots & & \vdots & & \vdots \\ 0 & 0 & \dots & 1 & \dots & 0 \\ \vdots & \vdots & & \vdots & & \vdots \\ 0 & 0 & \dots & \delta'_{n-1,k} & \dots & 1 \end{pmatrix},$$

la matrice obtenue suite à l'exécution de la deuxième boucle peut être retrouvée par

$$M_{\mathbf{s},0} = J'_0 \times M'_{\mathbf{s}}$$

Retrouver la matrice initiale

Pour le traitement de la colonne j, pour $n > j \geq 1$, nous considérons la matrice

$$M_{\mathbf{s},j-1} = \left(\prod_{k=j-1,\ldots,0} J'_k \times J_k \right) \times M_{\mathbf{s}}$$

Pour retrouver la colonne j, nous résolvons le système d'équations linéaires

$$J_j \times M_{\mathbf{s},j-1}[j] = (\delta'_j)^T$$

Sommaire

Contre-mesures

Comment protéger l'implantation de l'attaque proposée ?

Proposition de deux contre-mesures :

- Traitement aléatoire des coefficients dans chaque colonne de la matrice
  → un attaquant a $n!$ possibilités pour ordonner les éléments par colonne, soit $(n!)^m$

  ROLLO-I-128 → $(83!)^{67} \approx 2^{27731}$

- Ajout de masquage afin de ne plus distinguer de motifs dans la trace de consommation

Perspectives

ROLLO :

- ◇ Améliorer les performances et le coût mémoire de l'implantation avec les nouveaux paramètres
- ◇ Étude des vulnérabilités liées à l'implantation de la multiplication entre le chiffré et la clé privée
- ◇ Est-ce que l'attaque sur le pivot de Gauss peut s'appliquer à d'autres cryptosystèmes à base de codes correcteurs d'erreurs ?

Merci pour votre attention.

Références I

Carlos Aguilar-Melchor, Nicolas Aragon, Magali Bardet, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Alain Couvreur, Jean-Christophe Deneuville, Philipe Gaborit, Adrien Hauteville, and Gilles Zémor, *ROLLO-Rank-Ouroboros, LAKE & LOCKER*, 2019.

_____, *ROLLO-Rank-Ouroboros, LAKE & LOCKER*, 2020.

Magali Bardet, Pierre Briaud, Maxime Bros, Philippe Gaborit, Vincent Neiger, Olivier Ruatta, and Jean-Pierre Tillich, *An algebraic attack on rank metric code-based cryptosystems*, Advances in Cryptology – EUROCRYPT 2020, Springer International Publishing, 2020, pp. 64–93.

Philippe Delsarte, *Bilinear forms over a finite field, with applications to coding theory*, J. Comb. Theory, Ser. A **25** (1978), 226–241.

R. W. Hamming, *Error detecting and error correcting codes*, Bell System Technical Journal **29** (1950), no. 2, 147–160.