

CRYPTAGE PAR NON-INJECTION MASSIVE**Résumé :**

La problématique qui a conduit mes recherches est la suivante : construire un système de cryptage asymétrique qui s'affranchissant des "grands nombres premiers", utilise plutôt des "petits" nombres. Autrement dit que l'efficacité du cryptage ne soit pas due aux difficultés caché d'un objet simple et magique (comme les nombre premier ou les courbe elliptique qui pourrait réserver des surprises futurs), mais plutôt à des difficultés fabriqués manuellement à partir d'éléments simple reconnaissable comme essence de la difficulté technique (qui pourrait tout de même réserver des surprises futurs). L'idée est donc d'offrir une alternative. J'ai trouvé deux candidats technique pour mes réalisations qui donneront chacun lieu à une méthode sur F_p : les équations diophantiennes, les matrices non commutatives.

1 – Introduction : les principes utilisés**A - La non-injection**

Pour construire une fonction asymétrique, il faut soit qu'elle possède une très belle propriété (comme la commutativité de la fonction puissance) soit qu'elle soit irréversible sans espoir de retour. Dans la perspective de notre problématique, on comprend que la deuxième alternative est la plus probable. Mais alors il est inutile d'espérer pouvoir coder avec la même taille que le code initiale. On aura donc un allongement des données, mais c'est justement une des voie possible du cryptage (c'est notamment le principe fondamental de la stéganographie) :

Soit E, F deux ensembles finis tels que $\text{Card } F \gg \text{Card } E$

Soit C une fonction de cryptage de E dans F et D une fonction de décryptage de F dans E telles que $DoC=Id$

Le principe du codage consiste à disperser les valeurs de E dans F par un procédé qui ne permet pas de discerner où ces valeurs aboutissent, alors que la fonction D permet de les retrouver. L'idée consiste à « perdre dans le nombre », plutôt que dans la complexité d'une propriété particulière.

Comme application de ce principe, on peut par exemple partir d'un ensemble E et on crypte en arrivant dans un ensemble $F=E \times B$, le brouillage se fait en dispersant à l'aide de la dimension supplémentaire : B .

On peut discerner plusieurs pistes de mise en œuvre de ces idées.

B - le codage multiple

Soit $T \in E$ un texte à coder, Soient $(C_1(T), C_2(T), \dots, C_m(T)) \in F^m$ où les C_i sont des fonctions de codage qui sont publiques. Soit D une fonction de décodage (privée) telle que $D(C_1, C_2, \dots, C_m) = Id$

Il suffit donc que $\text{card}(F^m) \gg \text{Card}(E)$. Il pourrait être intéressant de rechercher de telles fonctions avec $\text{card } F < \text{card } E$, mais j'ai essentiellement travaillé sur le cas $E=F$. L'avantage est l'uniformité des calculs, mais le désavantage réside dans la taille du texte codé qui se trouve multiplié par m . Il va de soit qu'on prendra donc $m=2$ qui est suffisant pour un codage efficace, (le texte codé reste le double du texte original). Le choix de la méthode devra réaliser un compromis entre rapidité, sécurité, volume du texte crypté et volume des clés. Une application des plus simple de ce codage double est ce que j'appelle le « *Double Codage Composit* » :

On construit 3 fonctions C, D et R avec $DoC=R$.

Le cryptage d'un texte T sera fait par le couple $(Y_1, Y_2) = (C(T), (Id-R)(T))$ et le décryptage sera réalisé par $T = Y_2 + D(Y_1)$. La difficulté consiste alors à déterminer dans quel cadre mathématique construire des fonctions C, D et R telles que la connaissance de C et de R ne permet pas la découverte de D .

L'intérêt d'une telle méthode consiste en ce que D et C n'ont pas besoin d'être inversées et ainsi la complexité de la fonction ne pose aucun problème. Il me semble que c'est un atout non négligeable pour le cryptage. Les deux méthodes proposées par la suite seront basées sur ce *Double Codage Composit*.

C – Cacher par immersion dégradante.

Comme application du principe de non-injection, on peut imaginer une fonction de cryptage d'un espace E dans un espace F contenant E de telle sorte que E possède une structure forte avec beaucoup de propriétés, alors que F possède une structure beaucoup moins pratique, moins maniable ; ce qui permet de cacher les données (par exemple en utilisant la non-commutativité). Comme application de cette idée, on pourra voir les *Matrices Semi-commutatives* où le texte à crypter est fait de matrices qui commutent plongées dans un espace non commutatif.

D – Cacher par multiplicité

En travaillant sur le cryptage asymétrique dans des espaces commutatifs de petite dimension sur F_p , j'ai cru constaté que les méthodes simples ou immédiates sont presque inmanquablement « cassables » par résolution d'un système d'équations linéaires. En effet la commutativité implique très fortement la linéarité des équations. De cette remarque est issue l'idée suivante : pour cacher une information, on peut utiliser un espace de dimension sur F_p si importante que la résolution s'en trouve irréalisable non à cause de sa complexité mais par la taille du système linéaire à résoudre. Comme application de cette idée, on pourra voir la *Complexification Pyramidale*.

2 – Application : la complexification pyramidale**a- Données initiales**

On travaille dans F_p le corps à p éléments.

Soit $T=(t_i) \in (F_p)^n$ le texte à crypter ($i=1..n$).

On note P_p la F_p -algèbre des polynômes à une variable dans F_p .

(On constatera que tous les éléments de P_p ont un degré au plus égal à $p-1$ car dans F_p on a $X^p = X$)

b- Les polynômes inversibles

On aura besoin de 2 types de polynômes de P_p , décrivons les sommairement :

- Les polynômes inversibles pour la multiplication forment un ensemble qu'on notera I^M .

Si $f \in I^M$ alors il existe $f^{-1} / f^{-1} \cdot f = Id$
 Ce sont les polynômes f tels que $f(i) \neq 0$ pour tout i de F_p .

- Les polynômes inversibles pour la composition forment un ensemble qu'on notera de façon analogue I^C .
 Si $f \in I^C$ alors il existe $f^{-1} / f^{-1} \circ f = Id$
 Ce sont les polynômes qui établissent une bijection de F_p sur F_p .

[Pour fabriquer un quelconque de ces polynômes, il suffit de choisir une image b_i pour chaque élément i de F_p en respectant l'une des contraintes précédentes. Et pour trouver les coefficients du polynôme qui vont satisfaire à ces choix d'images, il suffit de résoudre le système linéaire formé par les p équations suivantes: $a_0 + a_1 \cdot i + a_2 \cdot i^2 + \dots + a_{p-1} \cdot i^{p-1} = b_i$]

c- Le schéma de base

Soit f, g, h dans P_p et x, y dans F_p . On appelle 'schéma de base' la fonction de $(F_p)^2$ dans F_p définie par $f(g(x)h(y))$. Cette fonction se développe en $\sum_{i,j=0..p-1} a_{i,j} x^i y^j$. L'idée de base consiste à dire que, si p est suffisant, il est "impossible" de

trouver f, g et h à partir de la connaissance des $a_{i,j}$. Cela reviendrait à résoudre un système du second degré trop complexe. [Si, contrairement à cette attente, la résolution n'est pas impossible, on peut compliquer ce schéma de base de bien des manières. Nous en donnerons un exemple plus loin].

L'idée consiste alors à combiner ce schéma de base de nombreuses fois afin de multiplier le nombre de variables de telles sorte que la fonction finale développée comporte tant de termes qu'il est impossible de l'utiliser sous sa forme développée.

d- Mise en œuvre

Le premier objectif consiste à construire une fonction de $(F_p)^n$ dans F_p .

Soit $T=(t_i)$ le texte à crypter $i=1..n$.

On prend $n=2^k$ pour construire une pyramide à k étages.

Soit $f_i \in I^M$ pour $i=1..2^k$; et $h_{1,i} \in I^C$ pour $i=1..2^{k-1}$.

On définit les $A_{1,i}$ (fonctions de $(F_p)^2$ dans F_p) à partir du schéma de base de la façon suivante :

$$A_{1,i}(t_{2i-1}; t_{2i}) = h_{1,i}(f_{2i-1}(t_{2i-1}) f_{2i}(t_{2i})) \text{ pour } i=1..2^{k-1}$$

Comme nous l'avons expliqué précédemment, la construction des $A_{1,i}$ est rendue publique sous forme développée alors que les fonctions f_j seront tenues secrètes.

On reproduit alors le phénomène en cascade (sous forme d'une pyramide) de la façon suivante :

Soit $h_{j,i} \in I^C$ pour $j=2..k$ et $i=1..2^{k-j}$, pour ces mêmes i et j on définit $A_{j,i}$ (fonctions $(F_p)^{2^j}$ dans F_p) de la façon suivante :

$$A_{j,i} = h_{j,i}(h_{j-1,2i-1}^{-1}(A_{j-1,2i-1}), h_{j-1,2i}^{-1}(A_{j-1,2i}))$$

qui sera donnée publiquement sous sa forme développée en les 2 variables : $A_{j-1,2i-1}$ et $A_{j-1,2i}$ (par les p^2 coefficients de F_p de cette forme développée).

Le calcul s'établit de façon étagée : à partir des t_i , on calcule les $A_{1,i}$.

A partir de ces derniers, on calcul les $A_{2,i}$ et ainsi de suite . De telle sorte qu'au dernier rang de la pyramide $A_{k,1}$ est une fonction dépendant de tous les t_i initiaux. On note $\tilde{A}_{k,1}$ cette fonction qui vérifie :

$$\tilde{A}_{k,1}(t_1, t_2, \dots, t_n) = A_{k,1}(A_{k-1,1}, A_{k-1,2}).$$

On s'apercevra que ces fonctions sont conçues de telles sortes que par simplification de la composition, on obtient :

$$\tilde{A}_{k,1}(t_1, t_2, \dots, t_n) = h_{k,1}(f_1(t_1) f_2(t_2) \dots f_n(t_n))$$

On note alors W l'application qui à $(f_1, f_2, \dots, f_n, h_{k,1})$ associe $\tilde{A}_{k,1}$ qui est une fonction de $(F_p)^n$ dans F_p

e- Première méthode

Par le procédé qui vient d'être décrit, on définit les fonctions C_j de $(F_p)^n$ dans F_p de la façon suivante :

Soit $f_{i,j} \in I^M$, $h_j \in I^C$ pour $i=1..n$, pour $j=1..n$, on construit

$$C_j = W(f_{1,j}, f_{2,j}, \dots, f_{n-1,j}, f_{n,j}, h_j)$$

On définit alors la fonction C de $(F_p)^n$ dans $(F_p)^n$ par

$$C(t_1, \dots, t_n) = (C_1(t_1, \dots, t_n), \dots, C_n(t_1, \dots, t_n))$$

Soit alors D_j pour $j=1..n$ des fonctions $(F_p)^n$ dans F_p définie par

$$D_j(u_1, u_2, \dots, u_n) = k_j([h_1^{-1}(u_1)]^{a_{1,j}} \dots [h_n^{-1}(u_n)]^{a_{n,j}}) \text{ où } a_{i,j} \in F_p \text{ et } k_j \in I^C.$$

En utilisant la construction des \tilde{A}_j , on voit qu'il existe $g_{i,j}$ des polynômes tels que :

$$D_j(C(t_1, \dots, t_n)) = k_j(g_{1,j}(t_1) g_{2,j}(t_2) \dots g_{n,j}(t_n)).$$

Comme les $f_{i,j}$ ne sont jamais nuls, il en est de même pour les $g_{i,j}$.

On définit $R_j = D_j \circ C$.

On voit qu'il est possible de directement construire ces R_j par une construction pyramidale semblable à celle des A_j de la forme $R_j = W(g_{1,j}, g_{2,j}, \dots, g_{n-1,j}, g_{n,j}, k_j)$ que l'on peut rendre publique de la même façon que les A_j par ses formes intermédiaires étagées et développées en deux variables.

Finalement, on obtient R , une deuxième fonctions de $(F_p)^n$ dans $(F_p)^n$ définie par

$$R(t_1, \dots, t_n) = (R_1(t_1, \dots, t_n), \dots, R_n(t_1, \dots, t_n)).$$

Ainsi les D_j qui sont la clé de décryptage sont gardés secrets.

La méthode de cryptage :

La clé de cryptage (C, R) donnée par les coefficients des polynômes développés à 2 variables permettant la construction pyramidale. Le Texte $T = (t_i)_{i=1..n}$ est crypté par

$$(\tilde{A}_1(T), \dots, \tilde{A}_n(T), t_1 - R_1(T), \dots, t_n - R_n(T))$$

La clé de décryptage sont les D_j car $D_j(\tilde{A}_1, \dots, \tilde{A}_n)(T) = R_j(T)$, il suffit de l'ajouter à $t_j - R_j(T)$ pour obtenir t_j .

Remarque :

Une des idée forte de ce cryptage est qu'en travaillant à partir des clés public les fonctions \tilde{A}_i et les fonctions R_i sont sensée être uniquement exprimable sous formes développé dans l'espace des fonctions de $(F_p)^n$ dans $(F_p)^n$. Or ces formes développées ne sont pas moins que la forme générale de tous les cryptages possible de $(F_p)^n$ dans $(F_p)^n$. Ainsi si la structures sous-jacente est réellement non traçable, il apparaît qu'on ne peut pas travailler dans un espace moins structuré. Il semble que de ce côté là le système soit bien protégé.

f- Deuxième méthode

Le principe général étant posé dans la première méthode, on peut imaginer beaucoup de variations utilisant ce même principe. On peut en particulier chercher à transformer la méthode pour faire des économies de calculs. Voici un exemple de ces transformations (qui est loin d'être optimisé, par ailleurs) :

Soit $e_i \in I^M$ pour $i=1..n$ et $f_i \in I^M$ pour $i=2..n$,

Par le procédé pyramidal, on construit

$$A_j = W(e_j, f_2, f_3, \dots, f_n, h_j) \text{ pour } j=1..n.$$

(les f_i sont fixes, seul e_j varie) avec

$$C(t_1, \dots, t_n) = (\tilde{A}_1(t_1, \dots, t_n), \dots, \tilde{A}_n(t_1, \dots, t_n)).$$

Soit alors D_j des fonctions définie par

$$D_j(u_1, \dots, u_n) = k_j(a_{1,j} \cdot h_1^{-1}(u_1) + \dots + a_{n,j} \cdot h_n^{-1}(u_n)) \text{ où les } a_{i,j} \text{ sont dans } F_p \text{ et } k_j \in I^C$$

Ainsi il existe $g_j \in I^M$ et R_j telles que

$$R_j(t_1, \dots, t_n) = D_j(C(t_1, \dots, t_n)) = k_j(g_j(t_1)f_2(t_2) \dots f_n(t_n)),$$

Et le cryptage s'effectue comme dans la première méthode.

g- Conclusion

On envisage deux axes possibles pour casser ce cryptage :

- Ou bien découvrir les clés en résolvant un système sur $(F_p)^n$ de degré strictement supérieur à 1 (le schéma de base). Nous avons supposé le système fiable de ce côté là.
- Ou bien découvrir directement une fonction qui permet de passer de C à R , le problème est alors linéaire, mais d'une telle taille qu'il est inaccessible.

[Si le premier point n'était pas satisfait, le procédé peut être compliqué de bien des façons. En voici un exemple :

Soit P_i une fonction polynomiale à deux variables dans F_p , pour $i=1..n$.

On pose $T_i = P_i(t_i, t_{i+1})$ et $T_n = P_n(t_n, t_1)$. Ces fonctions P_i resteront secrètes. On utilise cela pour transformer légèrement le schéma de base : on conserve la forme $f(g(T_i)h(T_{i+1}))$, mais au lieu d'être développé publiquement comme un polynome en 2 variables (T_i, T_{i+1}) , il le sera sous forme d'un polynomes en 3 variables (t_i, t_{i+1}, t_{i+2}) . Ce qui complique nettement le "braquage" de ce nouveau schéma de base ... Mais cela augmente aussi la taille des clés et le nombre de calculs.

En fait on pourra trouver de nombreuses autres méthodes basées sur le principe de la pyramide complexifiante. En particulier des méthodes qui utilise plusieurs schéma de bases différents. On peut aussi imaginer une stratégie de simplification globale au lieu d'être locale et identique à chaque niveau.]

En choisissant $n=16$ et $p=31$, il me semble qu'on peut trouver un niveau de sécurité tout à fait satisfaisant pour une rapidité très honorable dans le cadre d'une programmation software.

3- Application : Les matrices semi-commutatives.

a-Contexte

Tous les scalaires et tous les calculs seront réalisés dans F_p

Soit G l'espace des matrices carrés $r \times r$ avec $r=(n+m)$ à coefficients dans F_p ,

Si $e \in G$, on note :

e_1 la sous-matrice $n \times n$ située en haut à gauche de e , e_2 la sous-matrice $m \times n$ en haut à droite de e

e_3 la sous-matrice $n \times m$ en bas à gauche de e et e_4 la sous-matrice $m \times m$ en bas à droite de e .

On note (s,t) l'ensemble des matrices quelconques $M=(m_{i,j})$ telles que $m_{i,j}=0$ pour tout $i \neq t \pmod s$. La numération des indices i commençant à la valeur 0.

On note $[s,t]$ l'ensemble des matrices quelconques $M=(m_{i,j})$ telles que $m_{i,j}=0$ pour tout $j \neq t \pmod s$. La numération des indices j commençant à la valeur 0.

Soit A la partie de G telles que si $a \in A$ alors $a_1 \in [3,0]$; $a_2=0$; $a_3 \in [3,1]$ et $a_4=k.Id$, où k un élément de F_p .

Soit B la partie de G telles que si $b \in B$ alors $b_1 \in (3,0)$; $b_2 \in (3,2)$; $b_3=0$ et $b_4=k.Id$

Soit X l'ensemble des matrices qui constitue le texte à crypter, c' est un sous ensemble de G qui vérifie :

Si $x \in X$ alors $x_1=k.Id$, $x_2=0$, $x_3=0$, x_4 est quelconque

Cela signifie qu'un texte de base est constitué de m^2+1 éléments de F_p .

Soit C l'ensemble des matrices de G telles que si $c \in C$ alors c_1 est quelconque, $c_2=0$, $c_3=0$ et $c_4=k.Id$

Propriétés :

Si $a \in A$ et $b \in B$, alors $ab \in C$

Si $c \in C$ et $x \in X$, alors $cx=xc$ (C et X commutent)

Soit D l'ensemble des matrices de G telles que si $d \in D$ alors :

$d_1=d'+d''$ (matrices $n \times n$) avec $d' \in (3,0) \cap [3,0]$ et $d'' \in (3,1) \cap [3,1]$ d' et d'' étant des matrices $n \times n$

$d_2 \in (3,2)$, $d_3=0$ et $d_4=k.Id$

Soit E l'ensemble des matrices telles que si $e \in E$ alors :

$e_1=e'+e''$ (matrices $n \times n$) avec $e' \in (3,0) \cap [3,0]$ et $e'' \in (3,2) \cap [3,2]$ e' et e'' étant des matrices $n \times n$

$e_2=0$; $e_3 \in [3,3]$ et $e_4=k.Id$

Propriétés :

$a \in A$ et $d \in D$ alors $ad \in A$,

$e \in E$ et $b \in B$ alors $eb \in B$

b-Cryptage :

Soit $P(X) = \sum_{i=1}^n h_i X^i$ avec $h_i = b_i.c_i$ où $b_i \in B$ et $c_i \in C$

et $Q(X) = \sum_{j=0}^m K_j . (e_j . X . a . d)^j . L$ avec $K_j, L \in G$; $e_j \in E$; $a \in A$; $d \in D$

propriété :

En posant $R(X) = Q \circ P(X)$, à partir des précédentes propriétés, on peut montrer qu'il existe $U_i, V \in G$ tels que

$$R(X) = \sum_{i=0}^{mn} U_i . X^i . V$$

Le cryptage se fait donc par $C(X) = (P(X), X - R(X)) = (Y, Z)$. La clé publique sera donc (h_i, U_i, V)

Et le décryptage se fait par $D(Y, Z) = Z + Q(Y)$. La clé privée sera donc (K_j, e_j, a, d, L)

c-Commentaire sur la sécurité du système

On envisage trois axes possibles pour casser ce cryptage :

- Pour casser le code il suffit de trouver les clés privées : l'idée de base du cryptage consiste en ce que le développement de $Q \circ P$ peut se faire simplement. Car on a une semi-commutativité assurée par le fait que le produit $a.d.e_i.b_i.c_i \in C$. Comme C commute avec X , la simplification de l'expression peut se faire facilement. Par contre C n'étant pas lui-même commutatif, ce même développement ne peut pas se simplifier, il faut le calculer étape par étape. Ce qui rend très difficile la découverte des clés privées.
- Pour casser le code il suffirait de trouver une fonction $T(X)$ telles que $T(P(X)) = R(X)$, mais les matrices h_i n'étant pas commutative avec X , on ne peut calculer les puissances de $P(X)$ simplement. Il est nécessaire de multiplier $P(X)$ par une matrice qui rend le tout semi-commutatif pour pouvoir en calculer les puissances, mais on ne dispose d'aucune trace permettant de trouver celle qui a servi au cryptage. Et c'est là tout l'intérêt de la question. Ainsi trouver des matrices dont les puissances combinées vont donner U_i semble inaccessible...
- Il a fallu aussi préserver l'attaque directe sur X : en effet comme C et X sont commutatifs, il serait possible de restreindre la recherche aux transformations qui agissent sur X sans se soucier de la dimension inutile apportée par C , c'est pourquoi les matrices de B présentes dans h_i brouille les données de X et empêchent cette étude directe.

Le travail que je présente ici, n'est pas un travail achevé et mûri : de la théorie à la pratique, il y a beaucoup de subtilités, de possibilités, de choix qui devront être envisagés pour rendre la méthode retenue la plus sécurisée, la plus efficace et la plus légère possible. Mais il est inutile de se lancer dans un tel travail d'analyse avant de trouver confirmation aux bases théoriques. Voilà pourquoi je pose ici le point final.