Master 1 de Cryptologie et Sécurité Informatique
Équipe-projet GRACE

Stage estival 2017

Rapport de stage

Étude du cryptosystème de Chor-Rivest

Rémi Clarisse

Tuteurs : Daniel Augot et Luca De Feo

Introduction

En section 1, nous présentons l'Inria (Institut National de Recherche en Informatique et en Automatique), ainsi que l'équipe-projet GRACE. Les section 2 et section 3 exposent les travaux déjà effectués sur le cryptosysème de Chor-Rivest et la cryptanalyse de Vaudenay. La section 4 résume le papier de Joux sur le calcul de logarithmes discrets dans des corps finis de petite caractéristique, dont nous déduirons, en section 5, des améliorations possibles pour le cryptosystème.

1 La présentation de l'Inria

Inria emploie 2 600 collaborateurs issus des meilleures universités mondiales, qui relèvent les défis des sciences informatiques et mathématiques. Inria est organisé en «équipes-projets» qui rassemblent des chercheurs aux compétences complémentaires autour d'un projet scientifique focalisé. Ce modèle ouvert et agile lui permet d'explorer des voies originales avec ses partenaires industriels et académiques. Inria répond ainsi aux enjeux pluridisciplinaires et applicatifs de la transition numérique. A l'origine de nombreuses innovations créatrices de valeur et d'emploi, Inria transfère vers les entreprises (start-up, PME et grands groupes) ses résultats et ses compétences, dans des domaines tels que la santé, les transports, l'énergie, la communication, la sécurité et la protection de la vie privée, la ville intelligente, l'usine du futur ...

1.1 Centre Inria Saclay - Île-de-France

À Paris-Saclay, Inria développe des recherches à fort impact sociétal pour inventer le monde de demain. Créé en 2008, le centre de recherche Inria Saclay - Île-de-France accueille 450 scientifiques et 100 membres des services d'appui à la recherche. Les scientifiques sont organisés en 31 équipes de recherche dont 26 sont communes avec des partenaires du plateau de Saclay. Le centre accueille également le Joint Lab Inria / Microsoft Research.

«Le centre Inria Saclay - Île-de-France est un acteur essentiel de la recherche en sciences du numérique sur le plateau de Saclay. Il porte les valeurs et les projets qui font l'originalité d'Inria dans le paysage de la recherche : l'excellence scientifique, le transfert technologique, les partenariats pluridisciplinaires avec des établissements aux compétences complémentaires aux nôtres, afin de maximiser l'impact scientifique, économique et sociétal d'Inria.»
Bertrand Braunschweig, directeur du centre Inria Saclay - Île-de-France.

2 Le cryptosystème de Chor-Rivest

Dans leur article [1] publié en 1988, Benny Chor et Ronald Rivest introduisent un nouveau cryptosystème. Ils se sont inspiré du théorème de Bose-Chowla pour créer un cryptosystème reposant sur un problème de sac à dos et résistant à l'attaque Lagarias-Odlyzko [2].

Théorème 2.1 (Théorème de Bose-Chowla)**.** *Soient p un nombre premier et $h \geqslant 2$ un entier. Il existe une suite $(a_i)_{0 \leqslant i \leqslant p-1}$ d'entiers telle que :*

1. *pour tout $0 \leqslant i \leqslant p-1$,*

$$1 \leqslant a_i \leqslant p^h - 1,$$

2. *si* $(x_0, \ldots, x_{p-1})$ *et* $(y_0, \ldots, y_{p-1})$ *sont deux vecteurs distincts d'entiers naturels de poids inférieur à* h*, alors*

$$\sum_{i=0}^{p-1} x_i a_i \neq \sum_{i=0}^{p-1} y_i a_i.$$

2.1 Description

Soit p^h une puissance de nombre premier. Considérons le corps fini $\mathrm{GF}(p^h)$ dont il est supposé que sa représentation est publique, i.e. il existe un polynôme unitaire $P(x)$ public de degré h irréductible sur $\mathrm{GF}(p)$ et les éléments de $\mathrm{GF}(p^h)$ sont vus comme des polynômes modulo $P(x)$. Ainsi :

$$\mathrm{GF}(p^h) = \frac{\mathrm{GF}(p)[x]}{(P(x))}.$$

Notons $a = x \pmod{P(x)}$, la classe de x modulo $P(x)$ dans $\mathrm{GF}(p^h)$, et choisissons $(1, a, a^2, \ldots, a^{h-1})$ comme base du $\mathrm{GF}(p)$-espace vectoriel $\mathrm{GF}(p^h)$. Ainsi $\mathrm{GF}(p^h) = \mathrm{GF}(p)[a]$. Considérons aussi une numérotation publique α du sous-corps premier $\mathrm{GF}(p)$, i.e.

$$\{\alpha_0, \ldots, \alpha_{p-1}\} = \mathrm{GF}(p) \subset \mathrm{GF}(p^h).$$

Les éléments, choisis aléatoirement, de la clé privée – à garder secrète – consistent en :
— un élément $t \in \mathrm{GF}(p^h)$ algébrique de degré h sur $\mathrm{GF}(p)$, et on note $\mu(x) \in \mathrm{GF}(p)[x]$ le polynôme minimal de t sur $\mathrm{GF}(p)$,
— un générateur g du groupe cyclique $\mathrm{GF}(p^h)^\times$,
— un entier d tel que $0 \leqslant d \leqslant p^h - 2$,
— une permutation σ de l'ensemble $\{0, \ldots, p-1\}$.
La clé publique – à être publiée – est alors composée des :

$$c_i := d + \log_g \left(t + \alpha_{\sigma(i)}\right) \pmod{p^h - 1},$$

pour i tel que $0 \leqslant i \leqslant p-1$.

Étant donné qu'il faut calculer des logarithmes discrets pour fabriquer la clé publique, les paramètres choisis doivent permettre d'effectuer ces logarithmes facilement dans $\mathrm{GF}(p^h)$. Ainsi, Chor et Rivest suggèrent de prendre un entier premier, ou une puissance de nombre premier, p relativement petit et un exposant h friable, permettant l'utilisation de l'algorithme de Pohlig-Hellman. En l'occurrence, ils ont proposé de se placer dans les corps tels que $\mathrm{GF}(197^{24})$, $\mathrm{GF}(211^{24})$, $\mathrm{GF}(243^{24})$ et $\mathrm{GF}(256^{25})$.

L'espace des messages est l'ensemble des chaînes de p-bits de poids de Hamming égal à h. C'est-à-dire que le message à chiffrer doit d'abord être encodé en une chaîne de bits $m = [m_0 \cdots m_{p-1}]$ telle que $m_0 + \cdots + m_{p-1} = h$. N'importe quel message peut être découpé en plusieurs messages de p bits de poids h en suivant un algorithme d'encodage donné en section IV.B. de [1].

L'espace des chiffrés est $\mathbb{Z}/(p^h - 1)\mathbb{Z}$ et le chiffré d'un message m est :

$$E(m) := \sum_{i=0}^{p-1} m_i c_i \pmod{p^h - 1}.$$

Pour déchiffrer, nous calculons :

$$G(t) := g^{E(m) - hd}$$

vu comme un polynôme en t à coefficients dans $\mathrm{GF}(p)$ et de degré au plus $h-1$. Comme g est primitif dans $\mathrm{GF}(p^h)$, i.e. l'ordre de g est p^h-1, l'exposant $E(m)-hd$ est à déterminer modulo p^h-1 :

$$\begin{aligned} E(m)-hd &\equiv \left(\sum_{i=0}^{p-1} m_i c_i\right) - hd \pmod{p^h-1} \\ &\equiv \left(\sum_{i=0}^{p-1} m_i \left(d+\log_g\left(t+\alpha_{\sigma(i)}\right)\right)\right) - hd \pmod{p^h-1} \\ &\equiv \left(hd+\sum_{i=0}^{p-1} m_i \log_g\left(t+\alpha_{\sigma(i)}\right)\right) - hd \pmod{p^h-1} \\ &\equiv \sum_{i=0}^{p-1} m_i \log_g\left(t+\alpha_{\sigma(i)}\right) \pmod{p^h-1} \end{aligned}$$

D'où, l'égalité dans $\mathrm{GF}(p^h)$:

$$G(t) = g^{E(m)-hd} = \prod_{i=0}^{p-1}\left(t+\alpha_{\sigma(i)}\right)^{m_i} = \prod_{m_i=1}\left(t+\alpha_{\sigma(i)}\right).$$

Cela donne l'expression de l'élément $G(t)$ dans la base $(1,t,t^2,\ldots,t^{h-1})$ du $\mathrm{GF}(p)$-espace vectoriel $\mathrm{GF}(p^h)$, où $t = x \pmod{\mu(x)}$. Ainsi :

$$G(x) \equiv \prod_{m_i=1}\left(x+\alpha_{\sigma(i)}\right) \pmod{\mu(x)}.$$

Il existe donc un polynôme $\lambda(x) \in \mathrm{GF}(p)[x]$ tel que :

$$G(x) = \lambda(x)\mu(x) + \prod_{m_i=1}\left(x+\alpha_{\sigma(i)}\right).$$

En résonnant sur les degrés et du fait que $\mu(x)$ et $\prod\left(x+\alpha_{\sigma(i)}\right)^{m_i}$ soient unitaires, nous déduisons que $\lambda(x) = -1$, dont il découle l'égalité de polynômes :

$$G(x) + \mu(x) = \prod_{m_i=1}\left(x+\alpha_{\sigma(i)}\right).$$

La factorisation de ce dernier permet de recouvrer le message m.

2.2 Exemple

Prenons $p := 17$ et $h := 6$, ainsi $p^h = 24\,137\,569$.

```
sage: p = 17
sage: h = 6
sage: q = p ** h
```

Le corps fini $\mathrm{GF}(17^6)$ construit par `Sage` est, en l'occurrence,

$$\frac{\mathbb{F}_{17}[x]}{(P(x))}, \text{ où } P(x) := x^6 + 2x^4 + 10x^2 + 3x + 3 \in \mathbb{F}_{17}[x].$$

```
sage: K.<a> = FiniteField(q)
sage: P = a.minimal_polynomial()
```

La numérotation α du sous-corps premier choisie est :

$$(\alpha_0, \ldots, \alpha_{16}) = (2, 12, 4, 1, 0, 10, 7, 8, 15, 16, 3, 5, 13, 9, 11, 6, 14)$$

```
sage: alpha = [K(i) for i in range (p)]
sage: shuffle (alpha)
```

Génération des clés

Nous prenons un élément $t \in \mathrm{GF}(17^6)$ de degré algébrique 6 sur GF(17). Comme plus haut, nous notons $a = x \pmod{P(x)}$, le t sélectionné est alors :

$$t := 9a^5 + 16a^4 + 10a^3 + 3a^2 + 12a + 12,$$

de polynôme minimal :

$$\mu(x) := x^6 + 9x^5 + 8x^4 + 14x^3 + x^2 + 11x + 6 \in \mathrm{GF}(17)[x].$$

```
sage: while True :
sage:     t = K.random_element()
sage:     if t.minimal_polynomial().degree() == h :
sage:         break
sage: mu = t.minimal_polynomial()
```

Ensuite, comme élément primitif de $\mathrm{GF}(17^6)$, nous prenons :

$$g := 2a^5 + 5a^4 + 14a^3 + 2a^2 + 10a + 16.$$

```
sage: while True :
sage:     g = K.random_element()
sage:     if g.multiplicative_order() == q - 1 :
sage:         break
```

L'entier d est pris égal à 1 530 545.

```
sage: d = randint (0, q - 2)
```

Enfin, nous choisissons une permutation σ de l'ensemble $\{0, \ldots, 16\}$:

$$\sigma := (0, 10, 8, 5, 1, 6, 14)(2, 3, 9, 16, 13)(4, 12, 7, 15)(11).$$

```
sage: s = Permutations([i for i in range (p)]).random_element()
```

Nous avons fini de fabriquer la clé privée ! Reste à construire la clé publique :

c_0	:=	21 667 185	c_1	:=	3 210 064	c_2	:=	6 070 281
c_3	:=	3 093 929	c_4	:=	19 945 987	c_5	:=	294 610
c_6	:=	4 230 580	c_7	:=	18 951 770	c_8	:=	7 364 695
c_9	:=	23 348 812	c_{10}	:=	7 918 908	c_{11}	:=	3 562 855
c_{12}	:=	6 735 636	c_{13}	:=	13 077 876	c_{14}	:=	11 303 489
c_{15}	:=	22 106 426	c_{16}	:=	18 193 975			

```
sage: c = [mod (d + log (t + alpha[s[i]], g), q - 1)
                                              for i in range (p)]
```

Chiffrement d'un message

Maintenant, donnons nous un message à chiffrer de longueur 17 et de poids 6 :

$$m := [m_0 \cdots m_{16}] = [00100101100100100].$$

```
sage: m = [1 for i in range (h)] + [0 for i in range (p - h)]
sage: shuffle (m)
```

Chiffrons m :

$$e := 23\ 410\ 132.$$

```
sage: e = mod (sum ([m[i]*c[i] for i in range (p)]), q-1)
```

Déchiffrement du message

Nous souhaitons écrire g^{e-hd} comme un polynôme en t. Or `Sage` nous donne g^{e-hd} comme polynôme en a :

$$g^{e-hd} = a^5 + 11a^3 + 9a^2 + 15a + 1.$$

Pour parvenir à exprimer g comme nous le souhaitons, il faut effectuer un changement de base du GF(17)-espace vectoriel GF(17^6) : passer de la base $\mathcal{A} := (1, a, a^2, \ldots, a^{h-1})$ à la base $\mathcal{T} := (1, t, t^2, \ldots, t^{h-1})$. La matrice de passage facile à calculer est celle qui passe de la base $\mathcal{T}$ à la base $\mathcal{A}$: il suffit d'écrire dans `Sage` les différentes puissances de t, et `Sage` les exprime en fonction des puissances de a. Voici cette matrice :

$$\begin{bmatrix} 1 & 12 & 7 & 11 & 14 & 0 \\ 0 & 12 & 9 & 12 & 6 & 12 \\ 0 & 3 & 8 & 3 & 7 & 12 \\ 0 & 10 & 0 & 15 & 15 & 14 \\ 0 & 16 & 4 & 0 & 13 & 3 \\ 0 & 9 & 15 & 10 & 5 & 6 \end{bmatrix}$$

Celle qui nous intéresse est son inverse, la matrice de passage de la base $\mathcal{A}$ à la base $\mathcal{T}$:

$$\begin{bmatrix} 1 & 5 & 10 & 12 & 15 & 11 \\ 0 & 8 & 8 & 0 & 11 & 5 \\ 0 & 13 & 13 & 9 & 5 & 1 \\ 0 & 9 & 10 & 3 & 11 & 9 \\ 0 & 12 & 6 & 2 & 16 & 8 \\ 0 & 7 & 16 & 2 & 13 & 11 \end{bmatrix}$$

```
sage: V = K.vector_space()
sage: M = Matrix (GF(p),
                  [V(t ** i) for i in range (h)]).transpose()
sage: Minv = M.inverse()
```

Comme g^{e-hd} vaut dans la base $\mathcal{A}$ le vecteur $(1, 15, 9, 11, 0, 1)$, il est facile d'obtenir l'égalité :

$$g^{e-hd} = 10t^5 + 9t^4 + 12t^3 + 4t^2 + 10t + 3.$$

Notons $G(x) := 10x^5 + 9x^4 + 12x^3 + 4x^2 + 10x + 3 \in \mathrm{GF}(17)[x]$. Ainsi le message est recouvré en factorisant le polynôme :

$$G(x) + \mu(x) = x^6 + 2x^5 + 9x^3 + 5x^2 + 4x + 9,$$

ce que nous faisons :

$$G(x) + \mu(x) = (x+1)(x+2)(x+5)(x+6)(x+10)(x+12) \in \mathrm{GF}(17)[x].$$

Nous pouvons alors déterminer les α_i utilisés et nous savons alors que le bit $m_{\sigma^{-1}(i)}$ du message m est à 1, alors que les autres sont nuls.

```
sage: A.<x> = PolynomialRing (GF(p))
sage: Q = A(list (Minv * V(g ** (e - h * d)))) + A(mu)
sage: beta = [p - Q.roots()[i][0] for i in range (h)]
```

Pour facilité l'explication, notons β_i, pour i tel que $0 \leqslant i \leqslant 5$, les éléments de GF(17) tels que $-\beta_i$ est racine de $G(x) + \mu(x)$, les β_i étant tous distincts. Par exemple, $\beta_0 = 1$, $\beta_1 = 2$, …, $\beta_5 = 12$. Ainsi, pour tout i tel que $0 \leqslant i \leqslant 5$, il faut déterminer l'indice j, où $0 \leqslant j \leqslant 16$, tel que $\alpha_j = \beta_i$, et alors nous déduisons qu'il y a un 1 au $\sigma^{-1}(j)$ bit du message m. Par exemple, pour $\beta_0 = 1$: nous cherchons d'abord l'indice j tel que $\alpha_j = 1$, à savoir $j = 3$. Puis nous déterminons $\sigma^{-1}(3) = 2$. Et nous avons bien $m_2 = 1$. Idem, pour $\beta_2 = 5$: $\alpha_{11} = 5$, $\sigma^{-1}(11) = 11$ et nous constatons en effet que $m_{11} = 1$. Et ainsi de suite !

```
sage: sInv = [s.index(i) for i in range (p)]
sage: message = [0 for i in range (p)]
sage: for k in beta :
sage:     message[sInv [alpha.index(k)]] = 1
```

3 La cryptanalyse de Serge Vaudenay

Nous présentons ici l'attaque de Serge Vaudenay [3]. Elle s'appuie sur la forte friabilité de h, i.e. sur l'existence de nombreux sous-corps de $\mathrm{GF}(p^h)$. Pour casser une instance du cryptosystème, l'idée est de fabriquer une clé privée qui donne la même clé publique que le cryptosystème : cela va permettre de déchiffrer comme un destinataire légitime !

Ici, nous donnons l'attaque pour p un nombre premier, bien qu'elle soit généralisée pour le cas p puissance d'un nombre premier. Aussi, nous exposons l'attaque à rebours, à savoir qu'on part du cas où quelques éléments de la clé privée sont connus et au fur et à mesure, nous les retirons en regardant comment les fabriquer.

Symétrie dans la clé privée

Dans le cryptosystème de Chor-Rivest, nous choisissons la clé privée de façon aléatoire puis, à partir de celle-ci, nous calculons la clé publique. Le système repose sur la difficulté de trouver une clé secrète à partir de la clé publique. Cependant, nous remarquons qu'il y a plusieurs clés privées *équivalentes*, à savoir qu'il existe plusieurs jeux de clés privées qui donnent la même clé publique.

Affinons. Nous pouvons remplacer t et g par leur puissance p-ième, la clé publique reste inchangée car :

$$\log_{g^p}\left(t^p + \alpha_{\sigma(i)}\right) = \frac{1}{p}\log_g\left(\left(t + \alpha_{\sigma(i)}\right)^p\right) = \log_g\left(t + \alpha_{\sigma(i)}\right).$$

Nous pouvons aussi remplacer (t, α_σ) par $(t + v, \alpha_\sigma - v)$, pour tout $v \in \mathrm{GF}(p)$. Et enfin, nous pouvons remplacer (t, d, α_σ) par $(ut, d - \log_g(u), u\alpha_\sigma)$, quel que soit $u \in \mathrm{GF}(p)^\times$. Cela donne donc au moins $hp(p-1)$ clés privées équivalentes.

Il s'agit alors de déterminer une de ces clés.

Définition 3.1 (Clés équivalentes)**.** Deux clés privées sont dites *équivalentes* si elles fabriquent la même clé publique. En d'autres termes, deux clés privées (g, t, d, σ) et (g', t', d', π) sont équivalentes, si on a l'égalité d'ensembles :

$$\left\{d + \log_g(t + \alpha_{\sigma(i)}) : 0 \leqslant i \leqslant p-1\right\} = \left\{d' + \log_{g'}(t' + \alpha_{\pi(i)}) : 0 \leqslant i \leqslant p-1\right\}.$$

Pour passer d'une clé privée à une clé privée équivalente, il existe une transformation intéressante sur la permutation σ qui permet de choisir deux images arbitrairement pour deux antécédents fixés – ici, on prend les images de 0 et 1 :

Proposition 3.2. *Soit σ une permutation d'une clé privée. Alors, il existe π une permutation d'une clé privée équivalente, telle que,*

$$\pi(0) = i \qquad \textit{et} \qquad \pi(1) = j.$$

Démonstration. Posons :

$$u = \frac{\alpha_i - \alpha_j}{\alpha_{\sigma(0)} - \alpha_{\sigma(1)}} \in \mathrm{GF}(p)^\times \quad \text{et} \quad v = \frac{\alpha_j \alpha_{\sigma(0)} - \alpha_i \alpha_{\sigma(1)}}{\alpha_{\sigma(0)} - \alpha_{\sigma(1)}} \in \mathrm{GF}(p).$$

Il faut alors définir la permutation π par

$$\alpha_{\pi(i)} = u\alpha_{\sigma(i)} + v,$$

pour tout i tel que $0 \leqslant i \leqslant p-1$. □

Notation : Le discours qui suit porte sur la recherche de clés privées équivalentes. Souvent, elles sont exprimées en fonction de la clé privée possédée par le destinataire légitime. On note en lettre grasse la clé privée d'origine $(\mathbf{g}, \mathbf{t}, \sigma, \mathbf{d})$ et en police ordinaire une clé privée équivalente (g, t, π, d) – la permutation d'une clé équivalente est notée π.

Remarque 3.3. Aussi, nous faisons (parfois implicitement) le raccourci de « trouver une clé privée équivalente » à « trouver la clé privée équivalente ». Les algorithmes présentés donnent (quasiment) tous des listes d'éléments possibles en sortie, alors qu'un seul de ceux-ci nous suffit.

Cléf de voûte de l'attaque

Serge Vaudenay fabrique – il s'est appuyé sur ses travaux précédents – un polynôme bien particulier, qui lui sert de levier pour casser le cryptosystème. Le voici :

Proposition 3.4. *Pour tout facteur r de h, il existe un générateur $\mathbf{g}_{p^r}$ du groupe multiplicatif du sous-corps $\mathrm{GF}(p^r)$ de $\mathrm{GF}(p^h)$ et un polynôme $Q(x) \in \mathrm{GF}(p^r)[x]$ de degré h/r et tel que $-\mathbf{t}$ en est une racine, et pour tout $0 \leqslant i \leqslant p-1$, nous avons :*

$$Q\left(\alpha_{\sigma(i)}\right) = (\mathbf{g}_{p^r})^{c_i}.$$

Démonstration. Soit

$$Q(x) = (\mathbf{g}_{p^r})^{\mathbf{d}} \prod_{i=0}^{h/r-1} \left(x + \mathbf{t}^{p^{ri}}\right), \quad \text{où } \mathbf{g}_{p^r} := \prod_{i=0}^{h/r-1} \mathbf{g}^{p^{ri}},$$

$\mathbf{g}_{p^r}$ est la norme de $\mathbf{g}$ considéré dans l'extension de corps $\mathrm{GF}(p^r) \subseteq \mathrm{GF}(p^h)$, ainsi nous avons $(\mathbf{g}_{p^r})^{p^r} = \mathbf{g}_{p^r}$ et $\mathbf{g}_{p^r}$ est générateur de $\mathrm{GF}(p^r)^\times$ car $\mathbf{g}$ est primitif dans $\mathrm{GF}(p^h)$. Nous remarquons que $Q\left(x^{p^r}\right) = Q(x)^{p^r}$, ce qui prouve que $Q(x) \in \mathrm{GF}(p^r)[x]$. En effet :

$$\begin{aligned} Q\left(x\right)^{p^r} &= (\mathbf{g}_{p^r})^{\mathbf{d}p^r} \prod_{i=0}^{h/r-1} \left(x + \mathbf{t}^{p^{ri}}\right)^{p^r} \\ &= (\mathbf{g}_{p^r})^{\mathbf{d}} \prod_{i=0}^{h/r-1} \left(x^{p^r} + \mathbf{t}^{p^{ri}p^r}\right) \\ &= (\mathbf{g}_{p^r})^{\mathbf{d}} \prod_{i=1}^{h/r} \left(x^{p^r} + \mathbf{t}^{p^{ri}}\right) \\ &= Q\left(x^{p^r}\right), \end{aligned}$$

car $\mathbf{t}^{p^{r(h/r)}} = \mathbf{t}^{p^h} = \mathbf{t} = \mathbf{t}^{p^0}$. Cela montre au passage que $Q(-\mathbf{t}) = 0$. Aussi, $Q(x)$ est un polynôme de degré h/r. Et enfin, pour $0 \leqslant j \leqslant p-1$, calculons $(\mathbf{g}_{p^r})^{c_j}$:

$$\begin{aligned} (\mathbf{g}_{p^r})^{c_j} &= \left(\prod_{i=0}^{h/r-1} \mathbf{g}^{p^{ri}}\right)^{c_j} \\ &= \prod_{i=0}^{h/r-1} (\mathbf{g}^{c_j})^{p^{ri}} \\ &= \prod_{i=0}^{h/r-1} \mathbf{g}^{\mathbf{d}p^{ri}} \left(\alpha_{\sigma(j)} + \mathbf{t}\right)^{p^{ri}} \\ &= (\mathbf{g}_{p^r})^{\mathbf{d}} \prod_{i=0}^{h/r-1} \left(\alpha_{\sigma(j)} + \mathbf{t}^{p^{ri}}\right) \\ &= Q\left(\alpha_{\sigma(j)}\right) \end{aligned}$$

Cela achève la démonstration de la proposition. □

Comme h/r est assez petit, il est peu probable qu'il existe d'autres solutions $(\mathbf{g}_{p^r}, Q)$, et $\mathbf{g}_{p^r}$ est donc essentiellement unique. Soulignons la particularité de ce polynôme Q, il est à coefficients dans $\mathrm{GF}(p^r)$, il est de degré h/r et nous connaissons ses valeurs sur tout $\mathrm{GF}(p)$: il peut donc être interpolé sur un petit ensemble de $\mathrm{GF}(p)$.

Notation : Pour r divisant h et ω élément primitif de $\mathrm{GF}(p^h)$, notons

$$\omega_{p^r} := \omega^{(p^h-1)/(p^r-1)} = \omega^{1+p^r+p^{2r}+p^{3r}+\cdots+p^{h-r}},$$

la norme de ω considéré dans l'extension de corps $\mathrm{GF}(p^r) \subseteq \mathrm{GF}(p^h)$.

Attaque sachant $\mathbf{g}_{p^r}$ et σ

Supposons connus la norme $\mathbf{g}_{p^r}$ et la permutation σ. Nous pouvons alors interpoler le polynôme $Q(x)$ de la proposition 3.4, avec $h/r+1$ pairs $(\alpha_{\sigma(i)}, (\mathbf{g}_{p^r})^{c_i})_i$. Cela donne un

polynôme de degré h/r dont les racines sont les conjugués de $-\mathbf{t}$. Par la symétrie dans la clé privée, nous pouvons sélectionner n'importe quelle racine de $Q(x)$. Cet algorithme fait l'objet de la figure Algorithme 1.

Nous résolvons l'attaque en calculant g et d – d'une clé équivalente donc – grâce à l'attaque de Oded Goldreich exposée dans [1], une version simplifiée sera utilisée plus loin, c'est pour cela qu'elle est volontairement omise ici.

Algorithme 1 Algorithme implémentant l'attaque sachant $\mathbf{g}_{p^r}$ et σ

Entrée : $\mathrm{GF}(p^h)$, $(c_0, \ldots, c_{p-1})$, σ, r divisant h et $\mathbf{g}_{p^r}$
Sortie : $\mathbf{t}, \mathbf{t}^{p^r}, \mathbf{t}^{p^{2r}}, \ldots, \mathbf{t}^{p^{h-r}}$

1: $\rho \leftarrow \mathbf{g}_{p^r}$
2: $n \leftarrow h/r$
3: $Q(x) \in \mathrm{GF}(p^r)[x]$: initialisé $Q(x) \leftarrow 0$
4: **pour** i allant de 0 à n **faire**
5: calculer le polynôme interpolateur de Lagrange :

$$L(x) \leftarrow \prod_{\substack{0 \leqslant k \leqslant n \\ k \neq i}} \frac{x - \alpha_{\sigma(k)}}{\alpha_{\sigma(i)} - \alpha_{\sigma(k)}}$$

6: $Q(x) \leftarrow Q(x) + \rho^{c_i} L(x)$
7: **fin pour**
8: **retourner** $[-y \in \mathrm{GF}(p^h)$ pour y racine du polynôme $Q(x)]$

Simplification

Dans son papier, Serge Vaudenay interpole le polynôme $Q(x)$ de la proposition 3.4 sur un ensemble quelconque de $h/r + 1$ éléments de $\mathrm{GF}(p)$. À savoir, il choisit des indices $i_0, i_1, \ldots, i_{h/r}$ pris entre 0 et $p-1$, deux à deux distincts ; et interpole le polynôme en les éléments α_{i_j} pour j tel que $0 \leqslant j \leqslant h/r$.

Pour simplifier le propos, et surtout pour rendre plus lisible l'écriture, nous faisons le choix ici de prendre $i_j = j$, pour tout $0 \leqslant j \leqslant h/r$.

Attaque sachant $\mathbf{g}_{p^r}$

Supposons connu l'élément $\mathbf{g}_{p^r}$. Par la proposition 3.4, nous pouvons interpoler le polynôme $Q(x)$ en les $\alpha_{\sigma(j)}$, pour $0 \leqslant j \leqslant h/r$, grâce au polynôme interpolateur de Lagrange :

$$\begin{aligned} Q(x) &= \sum_{j=0}^{h/r} Q\left(\alpha_{\sigma(j)}\right) \prod_{\substack{0 \leqslant k \leqslant h/r \\ k \neq j}} \frac{x - \alpha_{\sigma(k)}}{\alpha_{\sigma(j)} - \alpha_{\sigma(k)}} \\ &= \sum_{j=0}^{h/r} (\mathbf{g}_{p^r})^{c_j} \prod_{\substack{0 \leqslant k \leqslant h/r \\ k \neq j}} \frac{x - \alpha_{\sigma(k)}}{\alpha_{\sigma(j)} - \alpha_{\sigma(k)}}, \end{aligned}$$

ce qui mène aux égalités : quel que soit i tel que $0 \leqslant i \leqslant p-1$,

$$(\dagger) \qquad (\mathbf{g}_{p^r})^{c_i} = \sum_{j=0}^{h/r} (\mathbf{g}_{p^r})^{c_j} \prod_{\substack{0 \leqslant k \leqslant h/r \\ k \neq j}} \frac{\alpha_{\sigma(i)} - \alpha_{\sigma(k)}}{\alpha_{\sigma(j)} - \alpha_{\sigma(k)}}.$$

En fait, nous pouvons même écrire : quel que soit i tel que $0 \leqslant i \leqslant p-1$,

$$(\ddagger) \qquad (\mathbf{g}_{p^r})^{c_i} - (\mathbf{g}_{p^r})^{c_0} = \sum_{j=1}^{h/r} ((\mathbf{g}_{p^r})^{c_j} - (\mathbf{g}_{p^r})^{c_0}) \prod_{\substack{0 \leqslant k \leqslant h/r \\ k \neq j}} \frac{\alpha_{\sigma(i)} - \alpha_{\sigma(k)}}{\alpha_{\sigma(j)} - \alpha_{\sigma(k)}}.$$

En effet, soit i tel que $0 \leqslant i \leqslant h/r$:

$$(\mathbf{g}_{p^r})^{c_i} - (\mathbf{g}_{p^r})^{c_0} = \sum_{j=1}^{h/r} ((\mathbf{g}_{p^r})^{c_j} - (\mathbf{g}_{p^r})^{c_0}) \prod_{\substack{0 \leqslant k \leqslant h/r \\ k \neq j}} \frac{\alpha_{\sigma(i)} - \alpha_{\sigma(k)}}{\alpha_{\sigma(j)} - \alpha_{\sigma(k)}},$$

car

$$\sum_{j=0}^{h/r} \left(\prod_{\substack{0 \leqslant k \leqslant h/r \\ k \neq j}} \frac{\alpha_{\sigma(i)} - \alpha_{\sigma(k)}}{\alpha_{\sigma(j)} - \alpha_{\sigma(k)}} \right) = 1.$$

Ainsi, nous avons l'égalité de polynôme :

$$Q(x) - (\mathbf{g}_{p^r})^{c_0} = \sum_{j=1}^{h/r} ((\mathbf{g}_{p^r})^{c_j} - (\mathbf{g}_{p^r})^{c_0}) \prod_{\substack{0 \leqslant k \leqslant h/r \\ k \neq j}} \frac{x - \alpha_{\sigma(k)}}{\alpha_{\sigma(j)} - \alpha_{\sigma(k)}},$$

car les deux polynômes sont de degré h/r et sont égaux sur un ensemble de $h/r + 1$ éléments : d'où l'égalité lorsque le polynôme est évalué en les $\alpha_{\sigma(i)}$, où $0 \leqslant i \leqslant p-1$.

À cause de la symétrie de la clé privée, nous pouvons choisir arbitrairement les images $\pi(0)$ et $\pi(1)$, où π est la permutation d'une clé équivalente à celle contenant σ. Un algorithme naïf pour trouver la permutation π est de chercher exhaustivement les valeurs $\pi(j)$ pour $2 \leqslant j \leqslant h/r$, jusqu'à ce que (†) donne une permutation consistante – avec σ substituée par π.

La complexité de cet algorithme est grossièrement $\mathcal{O}(rp^{h/r})$ opérations dans $\mathrm{GF}(p)$: la boucle à la ligne 4 fait en moyenne $\mathcal{O}(pr/h)$ itérations, chacune avec une complexité $\mathcal{O}(h)$, et il nous avons besoin de $\mathcal{O}(p^{h/r} - 1)$ itérations de cette boucle.

Par la proposition 3.2, la permutation π donnée par l'algorithme 2 est telle qu'il existe $u \in \mathrm{GF}(p)^\times$ et $v \in \mathrm{GF}(p)$ vérifiant, pour tout $0 \leqslant i \leqslant p-1$:

$$\alpha_{\pi(i)} = u\alpha_{\sigma(i)} + v.$$

Bien entendu, les éléments u et v sont liés à σ, ainsi, si σ est inconnu, alors en pratique, on ne peut pas calculer u et v. Cependant, cela ne nous arrête pas !

Algorithme 2 Algorithme pour trouver π sachant $\mathbf{g}_{p^r}$

Entrée : $\mathrm{GF}(p^h)$, $(c_0, \ldots, c_{p-1})$, r divisant h et $\mathbf{g}_{p^r}$
Sortie : une permutation π d'une clé privée équivalente

1: choisir arbitrairement $\pi(0)$ et $\pi(1)$ distincts dans $\{0, \ldots, p-1\}$
2: **pour tout** $\pi(2), \pi(3), \ldots, \pi(h/r)$ distincts deux à deux **faire**
3: $S \leftarrow [\pi(2), \pi(3), \ldots, \pi(h/r)]$
4: **pour tout** $\ell \notin S$ **faire**
5: calculer le membre de droite de (†) avec α_ℓ au lieu de $\alpha_{\pi(i)}$:

$$\mathbf{res} \leftarrow \sum_{j=0}^{h/r} (\mathbf{g}_{p^r})^{c_j} \prod_{\substack{0 \leqslant k \leqslant h/r \\ k \neq j}} \frac{\alpha_\ell - \alpha_{\pi(k)}}{\alpha_{\pi(j)} - \alpha_{\pi(k)}}$$

6: **si** il existe i tel que $\mathbf{res} = (\mathbf{g}_{p^r})^{c_i}$ et $\pi(i)$ n'est pas définie **alors**
7: $\pi(i) \leftarrow \ell$
8: $S \leftarrow S \cup [\pi(i)]$
9: **sinon**
10: continuer la boucle ligne 2
11: **fin si**
12: **fin pour**
13: **retourner** π
14: **fin pour**

En exprimant le polynôme $Q(x)$ faisant intervenir la permutation π par rapport au polynôme $Q(x)$ faisant intervenir la permutation σ, nous obtenons :

$$\begin{aligned}
Q_\pi(x) &= \sum_{j=0}^{h/r} (\mathbf{g}_{p^r})^{c_j} \prod_{\substack{0 \leqslant k \leqslant h/r \\ k \neq j}} \frac{x - \alpha_{\pi(k)}}{\alpha_{\pi(j)} - \alpha_{\pi(k)}}, \\
&= \sum_{j=0}^{h/r} (\mathbf{g}_{p^r})^{c_j} \prod_{\substack{0 \leqslant k \leqslant h/r \\ k \neq j}} \frac{x - u\alpha_{\sigma(k)} - v}{u\alpha_{\sigma(j)} - u\alpha_{\sigma(k)}}, \\
&= \sum_{j=0}^{h/r} (\mathbf{g}_{p^r})^{c_j} \prod_{\substack{0 \leqslant k \leqslant h/r \\ k \neq j}} \frac{u^{-1}x - u^{-1}v - \alpha_{\sigma(k)}}{\alpha_{\sigma(j)} - \alpha_{\sigma(k)}}, \\
&= Q_\sigma(u^{-1}(x - v)).
\end{aligned}$$

Ainsi, si $-\omega$ est racine du polynôme $Q_\pi(x)$, alors $-u^{-1}(\omega + v)$ est racine du polynôme $Q_\sigma(x)$, donc si on fait tourner l'algorithme 1 avec la permutation π, les valeurs renvoyées sont :

$$u\mathbf{t} - v, \quad u\mathbf{t}^{p^r} - v, \quad u\mathbf{t}^{p^{2r}} - v, \quad \ldots, \quad u\mathbf{t}^{p^{h-r}} - v.$$

Il existe alors un façon plus simple de déterminer g, que d'utiliser l'attaque de Goldreich (comme indiqué plus haut).

En effet : choisissons un générateur arbitraire γ de $\mathrm{GF}(p^h)^\times$, et pour un j tel que $0 \leqslant j < h/r$, à savoir on choisit n'importe quel élément retourné par l'algorithme 1,

calculons

$$b_i := \log_\gamma \left(u\mathbf{t}^{p^{jr}} - v + \alpha_{\pi(i)} \right)$$

Notons $L := \log_\gamma(\mathbf{g}^{p^{jr}})$, de sorte que $\mathbf{g}^{p^{jr}} = \gamma^L$. Alors pour tout $0 \leqslant i \leqslant p-1$:

$$\begin{aligned} b_i - b_0 &= \log_\gamma \left(\frac{u\mathbf{t}^{p^{jr}} - v + \alpha_{\pi(i)}}{u\mathbf{t}^{p^{jr}} - v + \alpha_{\pi(0)}} \right) \\ &= \log_\gamma \left(\frac{u\mathbf{t}^{p^{jr}} - v + u\alpha_{\sigma(i)} + v}{u\mathbf{t}^{p^{jr}} - v + u\alpha_{\sigma(0)} + v} \right) \\ &= \log_\gamma \left(\frac{u\mathbf{t}^{p^{jr}} + u\alpha_{\sigma(i)}}{u\mathbf{t}^{p^{jr}} + u\alpha_{\sigma(0)}} \right) \\ &= L \log_{\mathbf{g}^{p^{jr}}} \left(\frac{\mathbf{t}^{p^{jr}} + \alpha_{\sigma(i)}}{\mathbf{t}^{p^{jr}} + \alpha_{\sigma(0)}} \right) \\ &= L(c_i - c_0) \end{aligned}$$

Algorithme 3 Algorithme implémentant l'attaque de Goldreich simplifiée

Entrée : GF(p^h), $(c_0, \ldots, c_{p-1})$, t et π
Sortie : g et d

choisir γ un élément primitif arbitraire de GF(p^h)
calculer les $b_i := \log_\gamma(t + \alpha_{\pi(i)})$
$B \leftarrow [b_i - b_0$ pour i allant de 0 à $p-1]$
$C \leftarrow [c_i - c_0$ pour i allant de 0 à $p-1]$
pour i allant de 0 à $p-1$ **faire**
  si pgcd$(C[i], p^h - 1) = 1$ **alors**
    $L \leftarrow C[i]^{-1} B[i] \pmod{p^h - 1}$
    $\mathcal{C} \leftarrow \{L \cdot C[n] \pmod{p^h - 1}$ pour n allant de 0 à $p-1\}$
    $\mathcal{B} \leftarrow \{B[n] \pmod{p^h - 1}$ pour n allant de 0 à $p-1\}$
    si $\mathcal{C} = \mathcal{B}$ **alors**
      $g \leftarrow \gamma^L$
      $d \leftarrow c_0 - \log_g(t + \alpha_{\pi(0)}) \pmod{p^h - 2}$
      retourner g et d
    fin si
  fin si
fin pour

Bien sûr, si en entrée de l'algorithme 3, le t donné est $u\mathbf{t}^{p^{jr}} - v$, il retourne le conjugué $\mathbf{g}^{p^{jr}}$ du $\mathbf{g}$ d'origine et l'entier $\mathbf{d} - \log_{\mathbf{g}^{p^{jr}}}(u)$.

Cependant, quand r est suffisamment grand, il existe un meilleur algorithme pour trouver π. En fait, si $r \geqslant h/r$, alors la famille $((\mathbf{g}_{p^r})^{c_j} - (\mathbf{g}_{p^r})^{c_0})_{0 \leqslant j \leqslant h/r}$ est affinement indépendante. Cela signifie que les coefficients dans (†) sont les seuls coefficients dans GF(p) de l'écriture de $(\mathbf{g}_{p^r})^{c_i} - (\mathbf{g}_{p^r})^{c_0}$, pour $1 \leqslant i \leqslant p-1$, comme combinaison linéaire des vecteurs :

$$(\mathbf{g}_{p^r})^{c_1} - (\mathbf{g}_{p^r})^{c_0}, \quad \ldots, \quad (\mathbf{g}_{p^r})^{c_{h/r}} - (\mathbf{g}_{p^r})^{c_0}.$$

Notons a_j^i le coefficient de $(\mathbf{g}_{p^r})^{c_j} - (\mathbf{g}_{p^r})^{c_0}$ pour $(\mathbf{g}_{p^r})^{c_i} - (\mathbf{g}_{p^r})^{c_0}$, où $i, j \geqslant 1$. Alors, par (‡), nous avons, si $a_1^i \neq 0$:

$$\begin{aligned}
\frac{a_2^i}{a_1^i} &= \left(\prod_{\substack{0 \leqslant k \leqslant h/r \\ k \neq 2}} \frac{\alpha_{\sigma(i)} - \alpha_{\sigma(k)}}{\alpha_{\sigma(2)} - \alpha_{\sigma(k)}} \right) \left(\prod_{\substack{0 \leqslant k' \leqslant h/r \\ k' \neq 1}} \frac{\alpha_{\sigma(i)} - \alpha_{\sigma(k')}}{\alpha_{\sigma(1)} - \alpha_{\sigma(k')}} \right)^{-1} \\
&= \left(\prod_{\substack{0 \leqslant k \leqslant h/r \\ k \neq 2}} \frac{\alpha_{\sigma(i)} - \alpha_{\sigma(k)}}{\alpha_{\sigma(2)} - \alpha_{\sigma(k)}} \right) \left(\prod_{\substack{0 \leqslant k' \leqslant h/r \\ k' \neq 1}} \frac{\alpha_{\sigma(1)} - \alpha_{\sigma(k')}}{\alpha_{\sigma(i)} - \alpha_{\sigma(k')}} \right) \\
&= \left(\prod_{\substack{0 \leqslant k,k' \leqslant h/r \\ k \neq 2, k' \neq 1}} \frac{\alpha_{\sigma(1)} - \alpha_{\sigma(k')}}{\alpha_{\sigma(2)} - \alpha_{\sigma(k)}} \right) \left(\prod_{\substack{0 \leqslant k,k' \leqslant h/r \\ k \neq 2, k' \neq 1}} \frac{\alpha_{\sigma(i)} - \alpha_{\sigma(k)}}{\alpha_{\sigma(i)} - \alpha_{\sigma(k')}} \right) \\
&= \left(\prod_{\substack{0 \leqslant k,k' \leqslant h/r \\ k \neq 2, k' \neq 1}} \frac{\alpha_{\sigma(1)} - \alpha_{\sigma(k')}}{\alpha_{\sigma(2)} - \alpha_{\sigma(k)}} \right) \left(\frac{\alpha_{\sigma(i)} - \alpha_{\sigma(1)}}{\alpha_{\sigma(i)} - \alpha_{\sigma(2)}} \right)
\end{aligned}$$

Ainsi, il existe μ dans $\mathrm{GF}(p)$, indépendant de i, tel que

$$(\star) \qquad \frac{a_2^i}{a_1^i} = \mu \frac{\alpha_{\pi(i)} - \alpha_{\pi(1)}}{\alpha_{\pi(i)} - \alpha_{\pi(2)}} \quad \Leftrightarrow \quad \alpha_{\pi(i)} = \frac{a_2^i \alpha_{\pi(2)} - \mu a_1^i \alpha_{\pi(1)}}{a_2^i - \mu a_1^i}.$$

En passant toute les valeurs de μ en revue, nous pouvons obtenir $\pi(i)$ de l'équation $(\star)$. Cependant, cela ne permet pas de déterminer $\pi(0)$ et $\pi(j)$ pour $3 \leqslant j \leqslant h/r$. Il reste alors à les chercher de manière exhaustive. Cette remarque donne naissance à l'algorithme 4.

Algorithme 4 Algorithme pour trouver π sachant $\mathbf{g}_{p^r}$ lorsque $r \geqslant \sqrt{h}$

Entrée : $\mathrm{GF}(p^h)$, $(c_0, \ldots, c_{p-1})$, r divisant h tel que $r \geqslant \sqrt{h}$ et $\mathbf{g}_{p^r}$
Sortie : une permutation π d'une clé privée équivalente

1: pré-calculer la matrice de changement de bases de la "base classique" vers la "base" $((\mathbf{g}_{p^r})^{c_j} - (\mathbf{g}_{p^r})^{c_0})_{0 \leqslant j \leqslant h/r}$
2: choisir arbitrairement $\pi(1)$ et $\pi(2)$ distincts dans $\{0, \ldots, p-1\}$
3: **pour tout** μ dans $\mathrm{GF}(p)$ **faire**
4: **pour tout** i tel que $0 \leqslant i \leqslant p-1$ et $i \neq 0, 1, \ldots, h/r$ **faire**
5: écrire $(\mathbf{g}_{p^r})^{c_i} - (\mathbf{g}_{p^r})^{c_0}$ dans la "base" $((\mathbf{g}_{p^r})^{c_j} - (\mathbf{g}_{p^r})^{c_0})_{1 \leqslant j \leqslant h/r}$
6: récupérer les coefficients a_1^i et a_2^i
7: trouver la valeur de $\pi(i)$ grâce à $(\star)$
8: **fin pour**
9: **fin pour**
10: compléter π en cherchant de manière exhaustive les images $\pi(0), \pi(3), \ldots, \pi(h/r)$
11: **retourner** π

Test pour trouver $\mathbf{g}_{p^r}$

Les équations : quel que soit i tel que $0 \leqslant i \leqslant p-1$,

$$(\ddagger) \qquad (g_{p^r})^{c_i} - (g_{p^r})^{c_{i_0}} = \sum_{j=1}^{h/r} \left((g_{p^r})^{c_{i_j}} - (g_{p^r})^{c_{i_0}} \right) \prod_{\substack{0 \leqslant k \leqslant h/r \\ k \neq j}} \frac{\alpha_{\sigma(i)} - \alpha_{\sigma(i_k)}}{\alpha_{\sigma(i_j)} - \alpha_{\sigma(i_k)}},$$

signifient que tous les $(g_{p^r})^{c_i}$ sont en fait dans un même sous-espace affine de dimension h/r du $\mathrm{GF}(p)$-espace vectoriel $\mathrm{GF}(p^r)$.

Donc si on suppose que $h/r + 1 \leqslant r$, à savoir $r \geqslant \sqrt{h + 1/4} + 1/2$, on peut donner un test facile pour g_{p^r}.

Proposition 3.5. *S'il existe un facteur r de h de sorte que $r \geqslant \sqrt{h + 1/4} + 1/2$, et si*

$$g_{p^r} := g^{(p^h-1)/(p^r-1)} = g^{1+p^r+p^{2r}+\cdots+p^{h-r}},$$

alors tous les $g_{p^r}^{c_i}$ sont sur le même sous-espace affine de dimension h/r de $\mathrm{GF}(p^r)$ lorsque celui-ci est considéré comme un espace affine de dimension r sur $\mathrm{GF}(p)$.

Démonstration. To do!! □

L'existence d'un tel r peut être considérée comme un mauvais prérequis, cependant vu que les paramètres du cryptosystème de Chor-Rivest doivent être choisis de sorte que le problème du logarithme discret soit facile, nous savons déjà que h à plusieurs facteurs, et il est donc fort probable que cette hypothèse sur r soit satisfaite. En fait, les h sans tels facteurs sont les nombres premiers et les carrés de nombres premiers. Le vrai problème est que r ne doit pas être trop grand.

Nous pouvons donc écrire un algorithme qui vérifie si un candidat pour g_{p^r} est bon : l'algorithme vérifie simplement si les $g_{p^r}^{c_i}$ sont affinement liés. Cet algorithme a pour complexité moyenne $\mathcal{O}(h^3/r)$ opérations dans $\mathrm{GF}(p)$. Comme il y a $\phi(p^r-1)/r$ candidats, nous pouvons chercher exhaustivement un g_{p^r} avec une complexité de $\mathcal{O}(h^3 p^r / r^2)$.

Algorithme 5 Algorithme pour trouver g_{p^r} lorsque $r \geqslant \sqrt{h + 1/4} + 1/2$

Entrée : $\mathrm{GF}(p^h)$, $(c_0, \ldots, c_{p-1})$ et r divisant h tel que $r \geqslant \sqrt{h + 1/4} + 1/2$
Sortie : un élément g_{p^r}

1: choisir des $i_0, \ldots, i_{h/r}$ deux à deux distincts dans $\{0, \ldots, p-1\}$
2: **pour tout** $\zeta \in \mathrm{GF}(p^h)$ générateur de $\mathrm{GF}(p^r)^\times$ **faire**
3: calculer l'équation du sous-espace affine V engendré par $(\zeta^{c_{i_0}}, \ldots, \zeta^{c_{i_{h/r}}})$
4: **pour tout** i différent de $i_0, \ldots, i_{h/r}$ **faire**
5: **si** ζ^{c_i} ne satisfait pas à l'équation du sous-espace affine V **alors**
6: continuer la boucle ligne 2
7: **fin si**
8: **fin pour**
9: **retourner** ζ
10: **fin pour**

3.1 Utilisation des c_i

Nous allons améliorer l'attaque précédente en utilisant la connaissance de tous les c_i. Nous allons nous servir du fait suivant :

Proposition 3.6. *Soit $Q(x)$ un polynôme de $\mathrm{GF}(p^r)[x]$ de degré d et soit e un entier tel que $1 \leqslant e < (p-1)/d$. Nous avons*

$$\sum_{a \in \mathrm{GF}(p)} Q(a)^e = 0.$$

Le lemme qui suit va permettre de démontrer cette proposition.

Lemme 3.7. *Soit k un entier tel que $1 \leqslant k < p-1$. Alors*

$$\sum_{a \in \mathrm{GF}(p)} a^k = 0.$$

Ainsi, si $P(x) \in \mathrm{GF}(p^r)[x]$, pour $r \geqslant 1$, est de degré inférieur à $p-1$, alors :

$$\sum_{a \in \mathrm{GF}(p)} P(a) = 0.$$

Démonstration du lemme. Tout d'abord remarquons que $0 \in \mathrm{GF}(p)$ ne contribue pas à la somme, et ensuite, notons g un élément primitif du corps $\mathrm{GF}(p)$. Alors, pour $1 \leqslant k < p-1$:

$$\sum_{a \in \mathrm{GF}(p)} a^k = \sum_{a \in \mathrm{GF}(p)^\times} a^k = \sum_{j=0}^{p-2} g^{jk} = \frac{g^{k(p-1)} - 1}{g^k - 1} = \frac{1^k - 1}{g^k - 1} = 0.$$

Soit $P(x) \in \mathrm{GF}(p^r)[x]$ de degré inférieur à $p-1$:

$$P(x) = b_{p-2}x^{p-2} + \cdots + b_1 x + b_0, \qquad \text{où } b_i \in \mathrm{GF}(p^r).$$

Ainsi, on obtient bien :

$$\begin{aligned}
\sum_{a \in \mathrm{GF}(p)} P(a) &= \sum_{a \in \mathrm{GF}(p)} \left(b_{p-2}a^{p-2} + \cdots + b_1 a + b_0\right) \\
&= \sum_{k=1}^{p-2} \left(b_k \sum_{a \in \mathrm{GF}(p)} a^k \right) + p b_0 \\
&= 0.
\end{aligned}$$

□

Démonstration de la proposition. Comme $Q(x) \in \mathrm{GF}(p^r)[x]$ est de degré d et que $1 \leqslant e < (p-1)/d$, le polynôme $Q(x)^e$ est de degré inférieur à $p-1$. Ainsi le lemme précédent permet de conclure :

$$\sum_{a \in \mathrm{GF}(p)} Q(a)^e = 0.$$

□

Proposition 3.8. *Pour tout $1 \leqslant e < (p-1)r/h$, nous avons :*

$$\sum_{i=0}^{p-1} (g_{p^r})^{ec_i} = 0.$$

Démonstration. Considérons le polynôme $Q(x) \in \mathrm{GF}(p^r)[x]$ de la proposition 3.4 : le polynôme $Q(x)$ est de degré h/r, à coefficients dans $\mathrm{GF}(p^r)$ et tel que, pour tout $0 \leqslant i \leqslant p-1$:

$$Q\left(\alpha_{\sigma(i)}\right) = (g_{p^r})^{c_i}.$$

Ainsi par la proposition précédente, pour tout $1 \leqslant e < (p-1)r/h$:

$$\sum_{a \in \mathrm{GF}(p)} Q(a)^e = \sum_{i=0}^{p-1} Q\left(\alpha_{\sigma(i)}\right)^e = 0,$$

d'où l'égalité souhaitée :

$$\sum_{i=0}^{p-1} (g_{p^r})^{ec_i} = 0.$$

□

Cela donne une façon plus simple de sélectionner tous les candidats pour g_{p^r} – que la recherche exhaustive. Son principal avantage est qu'il fonctionne dans n'importe quel sous-corps. Par exemple, ont peut considérer $r = 1$ et trouver les seuls g_p tels que pour tout $1 \leqslant e < (p-1)/h$, nous avons :

$$\sum_{i=0}^{p-1} (g_p)^{ec_i} = 0.$$

La complexité moyenne pour vérifier un candidat est $\mathcal{O}(p)$ $\mathrm{GF}(p)$-opérations : il est peu probable qu'un mauvais candidat ne sera pas détecté par le $e = 1$. Ainsi, nous pouvons recouvrer g_p en $\mathcal{O}(p^2)$ opérations binaires.

Malheureusement, le g_{p^r} ne peut pas être utilisé efficacement quand r est trop petit. Cependant, on peut toujours utiliser g_{p^r} des petits sous-corps pour le calculer dans des corps plus grands. Notre but est de calculer g_{p^r} avec r assez grand (tel que $r^2 \geqslant h$). Nous considérons le problème du calcul de g_{p^r} lorsque $r_1, \ldots, r_k$ sont les facteurs de r et avec la connaissance de $g_{p^{r_i}}$. Comme $g_{p^{r_i}} = (g_{p^r})^{1+p^{r_i}+p^{2r_i}+\cdots+p^{r-r_i}}$, nous avons

$$(\dagger) \qquad \log(g_{p^r}) = \frac{\log(g_{p^{r_i}})}{1 + p^{r_i} + p^{2r_i} + p^{3r_i} + \cdots + p^{r-r_i}} \pmod{p^{r_i} - 1},$$

où la base des logarithmes est n'importe quel élément primitif γ fixé dans le corps $\mathrm{GF}(p^r)$. La connaissance de tous les $g_{p^{r_i}}$ donne donc la connaissance de $\log(g_{p^r})$ modulo

$$\ell := \mathrm{ppcm}\left\{p^{r_1} - 1, p^{r_2} - 1, \ldots, p^{r_k} - 1\right\}.$$

Nous avons donc besoin seulement de $(p^r - 1)/\ell$ essais pour recouvrer g_{p^r}.

Chaque boucle de la ligne 8 demande en moyenne $\mathcal{O}(pr^2)$ opérations dans $\mathrm{GF}(p)$.

Nous pouvons maintenant définir l'algorithme global de l'attaque.

Définition 3.9. Soit G un graphe orienté acyclique (*DAG for directed acyclic graph*) étiqueté enraciné dont la racine est étiquetée par un corps fini $\mathrm{GF}(p^r)$ et tel que pour chaque arête $u \to v$ de G, l'étiquette $l(u)$ de u est un sous-corps maximal (au sens de l'inclusion) de l'étiquette $L(v)$ de v et une extension de $\mathrm{GF}(p)$. On dit que G est un DAG p-factorisant de $\mathrm{GF}(p^r)$.

Algorithme 6 Algorithme pour récupérer g_{p^r} à partir des $g_{p^{r_i}}$

Entrée : $\mathrm{GF}(p^h)$, $(c_0, \ldots, c_{p-1})$, r divisant h, $\{r_i\}_{1 \leqslant i \leqslant k}$ diviseurs de r et $g_{p^{r_i}}$
Sortie : l'ensemble des g_{p^r} possibles

1: choisir γ un élément primitif de $\mathrm{GF}(p^r)$
2: **pour tout** i allant de 1 à k **faire**
3: résoudre l'équation (†) :

$$x_i = \frac{\log(g_{p^{r_i}})}{1 + p^{r_i} + p^{2r_i} + p^{3r_i} + \cdots + p^{r-r_i}} \pmod{p^{r_i} - 1}$$

4: **fin pour**
5: $\ell \leftarrow \mathrm{ppcm}\,\{p^{r_1} - 1, p^{r_2} - 1, \ldots, p^{r_k} - 1\}$
6: résoudre $x \equiv x_i \pmod{p^{r_i} - 1}$ où x est unique modulo ℓ
7: $\beta \leftarrow \gamma^x$
8: **pour** y allant de 0 à $(p^r - 1)/\ell - 1$ **faire**
9: **pour** e allant de 1 à $(p-1)r/h - 1$ **faire**
10: calculer la somme :

$$\mathbf{res} \leftarrow \sum_{i=0}^{p-1} \beta^{ec_i} \gamma^{ec_i \ell y}$$

11: **si res** $\neq 0$ **alors**
12: continuer la boucle ligne 8
13: **fin si**
14: **fin pour**
15: ajouter $\beta\gamma^{\ell y}$ à la liste des g_{p^r} possibles
16: **fin pour**

Définition 3.10. À une extension $\mathrm{GF}(p^r)$ et G son DAG p-factorisant, on associe :

$$C(G) := \sum_v \frac{\#L(v) - 1}{\mathrm{ppcm}\{\#L(w) - 1 \mid v \leftarrow w\}},$$

où nous prenons pour convention : $\mathrm{ppcm}(\emptyset) = 1$.

Nous pouvons définir un algorithme pour calculer g_{p^r} avec pour compléxité $\mathcal{O}(pr^2C(G))$, où G est le DAG p-factorisant de $\mathrm{GF}(p^r)$. Donc, nous pouvons casser le cryptosystème de Chor-Rivest avec comme paramètre h ni un nombre premier, ni un carré de nombre premier, avec une complexité de

$$\mathcal{O}\left(\min_{\substack{r \text{ divisant } h \\ r^2 \geqslant h}} \; \min_{\substack{G \text{ est un DAG} \\ p\text{-factorisant} \\ \text{de } \mathrm{GF}(p^r)}} pr^2 C(G) \right)$$

4 Le calcul de logarithmes discrets par Antoine Joux

5 Les conséquences pour le cryptosystème de Chor-Rivest

Algorithme 7 Attaque général

Entrée : GF(p^h) et $(c_0, \ldots, c_{p-1})$
Sortie : une clé secrète équivalente

1: pour le plus petit facteur r de h tel que $r \geqslant \sqrt{h + 1/4} + 1/2$, trouver G un DAG p-factorisant minimisant $C(G)$
2: **pour** u sommet de G, tel que pour tout $u \leftarrow u_i$, u_i a été visité, **faire**
3: l'algorithme 6 avec GF(p^r) = $L(u)$ et GF(p^{r_i}) = $L(u_i)$
4: **fin pour**
5: appliquer l'algorithme 4 à g_{p^r}

Références

[1] B. Chor and R. L. Rivest. **A Knapsack-Type Public Key Cryptosystem Based on Arithmetic in Finite Fields**. *IEEE Transactions on Information Theory*, 1988.

[2] J. C. Lagarias and A. M. Odlyzko. **Solving Low-Density Subset Sum Problems**. *IEEE Transactions on Information Theory*, 1983.

[3] S. Vaudenay. **Cryptanalysis of the Chor-Rivest Cryptosystem**. *Journal of Cryptology*, 2000.