Conception et preuves d'algorithmes cryptographiques

Cours de magistère M.M.F.A.I.
École normale supérieure

Jacques Stern
Louis Granboulan Phong Nguyen David Pointcheval

Édition 2004

Table des matières

1 Introduction à la cryptologie
— par Jacques Stern **5**
1.1 Qu'est ce que la cryptologie ? . . . 5
1.2 Cryptographie conventionnelle . . . 7
1.3 Méthodes statistiques de cryptanalyse . . . 10
1.4 Cryptographie à clé publique . . . 11
1.5 Cryptographie et complexité . . . 13

2 Chiffrement par bloc et cryptanalyse différentielle
— par Louis Granboulan **17**
2.1 Modes d'opération . . . 18
2.2 Principes de conception . . . 25
2.3 Étude théorique des schémas de Feistel . . . 29
2.4 Cryptanalyse différentielle . . . 30
2.5 Variantes de la cryptanalyse différentielle . . . 34
2.6 Autres techniques de cryptanalyse . . . 36

3 Vingt-cinq ans d'attaques de RSA
— par Phong Nguyen **43**
3.1 Le cryptosystème RSA . . . 44
3.2 RSA et la factorisation d'entiers . . . 46
3.3 Attaques élémentaires . . . 47
3.4 Attaques sur les implémentations du RSA . . . 50
3.5 Attaques simples du RSA à petit exposant . . . 51
3.6 Attaques à base de géométrie des nombres . . . 53
3. A L'algorithme LLL . . . 65

4 La sécurité prouvée en chiffrement asymétrique
— par David Pointcheval **71**
4.1 La cryptographie asymétrique . . . 72
4.2 Formalisation . . . 73
4.3 Les cryptosystèmes RSA et Rabin . . . 78
4.4 Le problème du logarithme discret . . . 80
4.5 Le cryptosystème de El Gamal . . . 82
4.6 Les attaques à chiffrés choisis adaptatives . . . 84

5 Zero-knowledge et identification
— par Jacques Stern **93**
5.1 Motivation et exemple . . . 93
5.2 Approche formelle du ZK . . . 96
5.3 Preuves ZK d'identité . . . 98

Chapitre 1 (2h)

Introduction à la cryptologie

— par Jacques Stern

Sommaire

1.1 Qu'est ce que la cryptologie ? 5
1.1.1 Quelques définitions 5
1.1.2 Quelques repères historiques 6
1.2 Cryptographie conventionnelle 7
1.2.1 Chiffrement et déchiffrement 7
1.2.2 Décryptement 7
1.2.3 Chiffrement par bloc 8
1.2.4 Chiffrement par flot 9
1.2.5 Intégrité et authenticité 9
1.3 Méthodes statistiques de cryptanalyse 10
1.3.1 Cryptanalyse du chiffrement de Vigenère 10
1.3.2 Cryptanalyse du chiffrement de Geffe 10
1.3.3 Tests d'hypothèse 11
1.4 Cryptographie à clé publique 11
1.4.1 Éléments de théorie algorithmique des nombres 11
1.4.2 RSA 12
1.5 Cryptographie et complexité 13
1.5.1 Machines de Turing polynomiales 13
1.5.2 Réduction et simulation 14

1.1 Qu'est ce que la cryptologie ?

1.1.1 Quelques définitions

La *cryptologie* est la science des messages secrets. Longtemps restreinte aux usages diplomatiques et militaires, elle est maintenant une discipline scientifique à part entière, dont l'objet est l'étude des méthodes permettant d'assurer les services d'intégrité, d'authenticité et de confidentialité dans les systèmes d'information et de communication.

Un service d'*intégrité* garantit que le contenu d'une communication ou d'un fichier n'a pas été modifié. Par exemple, on peut souhaiter vérifier qu'aucun changement du contenu

d'un disque dur n'a eu lieu : des produits commerciaux, mettant en jeu des méthodes cryptologiques, sont disponibles (voir notamment [12]) à cet effet.

Un service d'*authenticité* garantit l'identité d'une entité donnée ou l'origine d'une communication ou d'un fichier. Lorsqu'il s'agit d'un fichier et que l'entité qui l'a créé est la seule à avoir pu apporter la garantie d'authenticité, on parle de *non-répudiation*. Le service de non-répudiation est réalisé par une signature numérique. Une définition précise sera donnée plus loin ; on se bornera ici à constater que la loi du 20 mars 2000 [5] a fait passer ce concept dans la vie sociale.

Un service de *confidentialité* garantit que le contenu d'une communication ou d'un fichier n'est pas accessible aux tiers. Des services de confidentialité sont offerts dans de nombreux contextes

- en téléphonie mobile, pour protéger les communications dans la partie "aérienne" ;
- en télévision à péage pour réserver la réception des données aux abonnés ;
- dans les navigateurs, par l'intermédiaire du protocole SSL (Secure Socket Layer), dont l'activation est souvent indiquée par un cadenas fermé représenté en bas de la fenêtre.

La cryptologie se partage en deux sous-disciplines, également importantes : la *cryptographie* dont l'objet est de proposer des méthodes pour assurer les services définis plus haut et la *cryptanalyse* qui recherche des failles dans les mécanismes ainsi proposés.

1.1.2 Quelques repères historiques

L'ouvrage [10] distingue trois périodes dans l'histoire de la cryptologie. L'âge *artisanal* part des origines : Jules César utilisait, semble-t-il, un mécanisme de confidentialité rudimentaire, où chaque lettre d'un message était remplacée par celle située trois positions plus loin dans l'alphabet. La méthode se généralise et prend le nom de *substitution*. Une autre méthode change l'ordre des lettres ; elle a été mise en œuvre au Moyen-Âge notamment par un dispositif appelé "grille de Cardan". De façon générale, jusqu'au début du vingtième siècle, la cryptographie était affaire de substitutions et de transpositions. On opérait d'ailleurs fréquemment des substitutions non seulement sur des lettres mais sur des mots, en s'aidant d'une sorte de dictionnaire à double entrée, nommé *code* ou *répertoire*. La cryptanalyse, quant à elle, utilisait des méthodes statistiques simples, fondées principalement sur la fréquence des lettres ou des suites de deux lettres (digrammes) dans un texte.

L'âge *technique* garde les substitutions et les permutations mais les met en œuvre à l'aide de machines mécaniques ou électro-mécaniques. Les plus célèbres sont la Hagelin et l'Enigma utilisée par l'armée allemande durant la seconde guerre mondiale. La complexité des méthodes rendues ainsi accessibles étant plus grande, la cryptanalyse devient plus conceptuelle et a aussi recours à des machines. Pour venir à bout de l'Enigma, les Alliés réunissent à Bletchley Park un groupe de scientifiques, dont Alan Turing, inventeur des machines qui portent son nom (et que nous retrouverons plus loin). Turing parvient à réaliser une spectaculaire cryptanalyse en la réduisant à une recherche de cas suffisamment restreinte pour être menée par une machine spécialement construite à cet effet. C'est aussi Turing qui, dans le cadre d'une autre cryptanalyse également réussie, fit construire le Colossus, doté d'électronique, et qui peut être considéré comme l'un des ancêtres de nos ordinateurs modernes.

L'âge *paradoxal* couvre les vingt-cinq dernières années. Il voit l'introduction de mécanismes donnant des réponses positives à des questions *a priori* hors d'atteinte :

- Comment assurer un service de confidentialité sans avoir au préalable établi une convention secrète commune ?

– Comment assurer un service d'authenticité — basé sur la possession d'un secret — sans révéler la moindre information sur le secret ?

La période récente est également marquée par le développement d'une importante communauté de recherche. Cette communauté a largement transformé l'image de la cryptologie : elle a apporté une plus grande rigueur a la cryptographie en essayant de produire, autant que possible des preuves partielles de sécurité, de type mathématique. Elle a également donné un statut nouveau à la cryptanalyse, destinée maintenant à valider les méthodes proposées par une approche systématique, plutôt qu'à donner un avantage compétitif ou stratégique.

1.2 Cryptographie conventionnelle

1.2.1 Chiffrement et déchiffrement

La cryptographie conventionnelle est principalement liée aux services de confidentialité. Elle réalise sur les données m une transformation $c = \mathsf{E}_\mathsf{k}(m)$, par l'intermédiaire d'un algorithme de *chiffrement* E. Cet algorithme prend en entrée le message clair m et un paramètre secret k, qu'on appelle la *clé*. Le message m varie dans un ensemble $\mathcal{M}$ et la clé k dans un ensemble $\mathcal{K}$. La restauration du texte clair à partir du *chiffré* ou *cryptogramme* c se fait par un algorithme de *déchiffrement* D_k, prenant en entrée le chiffré et la même clé. On doit avoir $\mathsf{D}_\mathsf{k}(\mathsf{E}_\mathsf{k}(m)) = m$. En général, le chiffré prend sa valeur dans le même espace $\mathcal{M}$ et l'on a aussi $\mathsf{E}_\mathsf{k}(\mathsf{D}_\mathsf{k}(c)) = c$, c'est à dire que les algorithmes E_k et D_k réalisent une permutation de $\mathcal{M}$.

La distinction entre l'algorithme et la clé s'est établie il y a fort longtemps, notamment dans les travaux du cryptologue Auguste Kerckhoffs [4]. Ce dernier a en effet su reconnaître que l'algorithme de chiffrement n'exigeait pas le secret, dans la mesure où il risquait de toutes façons de passer aux mains de l'ennemi. La cryptologie moderne recommande même des méthodes de chiffrement totalement explicites, de manière à ce qu'elles soient évaluées et validées par un débat ouvert entre experts. Du coup, une convention secrète entre entités qui souhaitent communiquer de façon chiffrée, se limite à l'échange d'une clé k.

1.2.2 Décryptement

L'opération qui consiste à calculer le clair m à partir du chiffré $c = \mathsf{E}_\mathsf{k}(m)$, mais sans la connaissance de la clé k est appelée *décryptement*. La confidentialité est assurée si cette opération est impossible. On distingue divers scénarios possibles d'attaque

– les attaques à chiffré seul, où l'adversaire dispose d'un certain nombre de chiffrés $\mathsf{E}_\mathsf{k}(m_i)$;
– les attaques à clair connu, où l'adversaire dispose d'un certain nombre de chiffrés $\mathsf{E}_\mathsf{k}(m_i)$ et des clairs correspondants m_i ;
– les attaques à clair choisi, où l'adversaire dispose d'un certain nombre de chiffrés $\mathsf{E}_\mathsf{k}(m_i)$ correspondant à des clairs de son choix m_i ; si de plus chaque message m_i est défini en fonction des chiffrés obtenus antérieurement, on parle d'attaque à clair choisi adaptative.

Le lecteur pourra définir d'autres variantes, comme l'attaque à chiffré choisi. Le but de l'attaque est la découverte de la clé ou le décryptement d'un chiffré c, correspondant à un clair dont on ne dispose pas. Les attaques à chiffré seul sont les plus difficiles. Néanmoins, l'adversaire dispose en général d'informations statistiques sur le clair. En d'autres termes,

les messages sont créés en suivant une probabilité qui correspond à une distribution sur $\mathcal{M}$, appelée distribution *a priori*. L'interception d'un (ou plusieurs) chiffrés a pour effet de conditionner cette distribution, produisant une distribution *a posteriori* : par exemple si l'on sait qu'un message chiffré provient d'une distribution équiprobable sur les mots "tas", "sas", "mur" et si le chiffrement est une substitution de lettres, alors l'interception du chiffré XUV élimine "sas". On dit qu'un chiffrement est parfait si les deux distributions coïncident. Le théorème de Shannon énonce que l'espace des clés $\mathcal{K}$ est alors de taille au moins égale à l'espace des messages. Il existe d'ailleurs un mécanisme appelé chiffrement de Vernam ou *one-time pad*, qui assure un tel niveau de sécurité : il consiste à chiffrer un message de b bits m_i à l'aide d'un clé k de b bits également, le chiffré étant le "ou exclusif bit-à-bit" défini par $c_i = m_i \oplus \mathsf{k}_i$. Pour autant qu'on génère la clé aléatoirement et qu'on ne l'utilise qu'une fois, cette méthode de chiffrement offre une sécurité absolue, qu'on nomme aussi *inconditionnelle*.

En général, on ne peut utiliser un chiffrement de Vernam et on conserve une même clé k pour chiffrer un certain nombre de messages. La connaissance d'un petit nombre de chiffrés produit alors une distribution conditionnelle qui définit la clé de manière unique. Pour le comprendre, il suffit d'imaginer qu'un algorithme de chiffrement opère sur des mots de huit octets mais qu'on a intercepté quelques chiffrés correspondant à des suites de huit caractères ASCII 7 bits. Pour chaque clé k et pour chaque chiffré intercepté c, la probabilité que $\mathsf{D}_\mathsf{k}(c)$ soit un message bien formé est environ 1/2000. Ce chiffre provient, par un calcul simple, du pourcentage dans chaque octet des caractères ASCII, lequel est de 38.6 %. Si donc on exploite 10 chiffrés l'espace des clés compatibles avec ces chiffrés est réduit d'un facteur environ 2^{-110}. Même pour des clés de 128 bits, on n'a plus que quelques solutions et on tombe rapidement à une seule solution avec quelques chiffrés supplémentaires. De fait la sécurité devient *algorithmique* : on ne peut que demander que, compte tenu de la puissance de calcul dont il dispose, l'adversaire ne puisse déterminer l'unique valeur de la clé. A cet égard, il existe toujours une méthode permettant de retrouver la clé à partir de quelques couples clair/chiffré, (m_i, c_i), en nombre suffisant pour assurer l'unicité. Elle consiste à explorer l'espace des clés et à tester pour chaque clé si $\mathsf{E}_\mathsf{k}(m_1) = c_1$. Si le test est positif, on effectue le teste analogue sur m_2 et ainsi de suite. On s'arrête quand la clé a été trouvée. En moyenne, on parcourt la moitié de l'espace des clés $\mathcal{K}$.

1.2.3 Chiffrement par bloc

Dans les algorithmes de chiffrement par *bloc*, l'espace des messages est de la forme $\{0,1\}^b$. Autrement dit le clair (comme le chiffré) est une suite de b bits. Des messages de taille supérieure à b sont chiffrés en les complétant à un multiple de b bits, par une règle de formatage convenue et en chiffrant bloc par bloc. Il existe plusieurs modes d'opération. Le mode ECB (*electronic code book*) chiffre successivement chaque bloc. Le mode CBC (*cipher block chaining*), fait le "ou exclusif" de chaque bloc avec le chiffré précédent avant d'appliquer l'algorithme de chiffrement, soit $c_i = \mathsf{E}_\mathsf{k}(c_{i-1} \oplus m_i)$. On peut convenir que, pour chiffrer le premier bloc m_1, on prend c_0 nul ou ajouter un vecteur d'initialisation IV, transmis en clair, et poser $c_0 = IV$. Le déchiffrement calcule m_i par $c_{i-1} \oplus \mathsf{D}_\mathsf{k}(c_i)$.

Le plus connu des algorithmes de chiffrement est le DES (voir [8, 6, 11]). Il opère sur des blocs de 64 bits avec des clés de 56 bits. Il est essentiellement composé d'une suite de 16 tours identiques, chaque tour réalisant une transformation de Feistel. Une telle transformation génère une permutation sur $2n$ bits à partir d'une fonction f dépendant d'une clé k et dont les entrées sont sur n bits. Les $2n$ bits sont séparés en deux blocs de n bits L et R et on

pose $L' = R$, $R' = L \oplus f_{\mathsf{k}}(R)$. Cette fonction est inversible. Les clés de tour sont formés de 48 bits extraits de la clé du DES par une méthode qui dépend du tour considéré. Une description plus précise du DES sera fournie ultérieurement. On considère aujourd'hui le DES comme obsolète, principalement à cause de la taille trop réduite de la clé. On utilise souvent le triple DES avec deux clés $(\mathsf{k}_1, \mathsf{k}_2)$, la fonction de chiffrement étant dérivée de celle du DES par la formule $\mathsf{E}_{\mathsf{k}_1}(\mathsf{D}_{\mathsf{k}_2}(\mathsf{E}_{\mathsf{k}_1}(m)))$. En prenant $\mathsf{k}_1 = \mathsf{k}_2$, on retrouve le DES.

Le successeur officiel du DES est l'AES [1], choisi après une compétition ouverte aux équipes de recherche industrielles et académiques. C'est un algorithme de chiffrement par blocs dont les blocs ont 128 bits et les clés ont 128, 192 ou 256 bits. L'AES est une suite de r tours, chacun réalisant une suite de permutations et de substitutions dépendant d'une clé de tour et opérant sur une matrice 4×4 d'octets. La valeur de r est fixée à 10 pour les clés de 128 bits, à 12 pour des clés de 192 bits et à 14 pour des clés de 256 bits.

1.2.4 Chiffrement par flot

Dans les algorithmes de chiffrement par *flot*, une suite d'octets ou de bits r_i est produite à partir de la clé. Cette suite est combinée aux octets ou aux bits du clair m_i pour donner les octets ou les bits du chiffré c_i, suivant la formule $c_i = m_i \oplus r_i$.

RC4 est un algorithme de chiffrement par flot, utilisé notamment dans le protocole SSL de Netscape. C'est la propriété de la société RSA Data Security Inc. mais les versions publiées, par exemple dans [8], n'ont pas été démenties. A partir de la clé de longueur variable, par exemple 128 bits, un tableau S de 256 octets est initialisé et deux compteurs i et j mis à zéro. Pour générer un nouvel octet aléatoire, on applique les opérations suivantes

$i = (i + 1) \bmod 256$
$j = j + S[i] \bmod 256$
échanger $S[i]$ et $S[j]$
$t = S[i] + S[j] \bmod 256$
retourner $S[t]$

Une méthode extrêmement efficace pour produire une suite de bits utilisable pour un chiffrement par flot, notamment dans les environnements matériels se fonde sur les registres à décalages. Ces dispositifs ont L registres, numérotés de 0 à $L - 1$, chacun contenant un bit d'état interne. A chaque coup d'horloge, le contenu du registre numéroté 0 est retourné, le contenu s_i du i-ième registre $(i \geq 1)$ avance dans le $i - 1$-ième. Le dernier registre s_{L-1} reçoit une valeur calculée par une fonction de rétroaction f dépendant de $s_{L-1}, \cdots s_0$, notée $f(s_{L-1}, \cdots s_0)$. Il est clair que si le contenu initial des registres est $[s_{L-1}, \cdots, s_0]$, le bit s_j produit au j-ième coup d'horloge est donné, pour $j \geq L$, par la relation de récurrence

$$s_j = f(s_{j-1}, s_{j-2}, \cdots, s_{j-L})$$

Lorsque f est linéaire on parle de registre à décalages linéaire (LFSR, linear feedback shift register).

1.2.5 Intégrité et authenticité

Le service d'intégrité est assuré par un algorithme qu'on peut qualifier de conventionnel même si les définitions qui suivent sont récentes. Une *fonction de hachage cryptographique* H calcule un condensé de taille fixe à partir d'un message formé d'une suite de bits quelconque. On requiert qu'il soit pratiquement impossible à un adversaire de calculer des *collisions*,

c'est à dire de produire des messages m et m' différentes mais tels que $H(m) = H(m')$. Cette propriété empêche la substitution d'un message à un autre, si le condensé est conservé séparément. La fonction de hachage la plus répandue est la norme SHA-1. Elle produit des condensés de 20 octets.

Les méthodes conventionnelles assurent des services d'authenticité, mais ne garantissent pas la non-répudiation, puisque les clés secrètes sont partagées. Pour authentifier un message, sans le chiffrer, on peut calculer et transmettre le dernier chiffré dans un chiffrement CBC du message, voire une partie de ce chiffré. Le nom traditionnel de cette méthode est CBC-MAC, le mot MAC étant l'acronyme de *Message Authentication Code*.

1.3 Méthodes statistiques de cryptanalyse

On va donner ici quelques exemples d'utilisation des probabilités et statistiques en cryptanalyse. Ces exemples seront complétés dans la suite du cours. On verra également dans la suite du cours que la cryptologie de l'âge paradoxal s'est enrichie de méthodes de cryptanalyse plus algébriques.

1.3.1 Cryptanalyse du chiffrement de Vigenère

Pour mesurer la distance entre deux distributions de probabilités $\mathcal{D}_1$ et $\mathcal{D}_2$ sur un même espace de probabilité fini, on peut utiliser leur distance, définie par

$$\sum_x |\mathsf{pr}_1(x) - \mathsf{pr}_2(x)|$$

où x décrit l'espace et pr_i désigne la probabilité relative à $\mathcal{D}_i$. On peut aussi utiliser leur distance euclidienne

$$(\sum_x |\mathsf{pr}_1(x) - \mathsf{pr}_2(x)|^2)^{1/2}$$

Le carré de la distance euclidienne de la distribution des lettres dans une langue donnée à la distribution uniforme est un invariant κ qui vaut 0.0393 en français et 0.0282 en anglais. Considérons un algorithme qui effectue sur un texte d'une des langues une suite périodique de transformations, chacune réalisant une substitution fixe de lettres, les substitutions étant choisies indépendemment les unes des autres. Un tel algorithme est appelé chiffrement de Vigenère et sa période t est supposée inconnue. La probabilité que deux occurrences de deux lettres du cryptogramme coïncident est de $\sum_{i=1}^{n} p_i^2$, lorsque les occurrences sont à distance multiple de t et $1/n$ autrement. Dans ce qui précède, p_i désigne la probabilité d'apparition (dans le clair) de la i-ème lettre et n le nombre de lettres. La différence est exactement κ et on peut ainsi retrouver la valeur secrète de t en calculant pour $t = 1, 2, \ldots$, la probabilité que $c_i = c_{t+1}$ dans le texte chiffré. Un pic apparaît pour la bonne valeur de t.

1.3.2 Cryptanalyse du chiffrement de Geffe

L'algorithme de Geffe est un algorithme de chiffrement par flot qui combine les sorties de trois LFSR, soit x_j, y_j, z_j par la fonction booléenne (multiplexeur) $z_j x_j \oplus (1 \oplus z_j) y_j$. A priori, la clé secrète du générateur se compose du contenu initial des trois LFSR. Toutefois, on observe que la j-ème sortie du générateur de Geffe vaut x_j avec probabilité 3/4. Supposons maintenant que le texte clair soit une suite de caractères ASCII 7 bits. Alors, en extrayant le

bit de chaque octet dont le clair vaut zéro, on obtient une suite de bits égaux avec probabilité 3/4 au bit correspondant de x_j. On peut donc tester chaque état initial du premier LFSR, calculer la suite x_j correspondante et évaluer la fréquence de coïncidence avec x_j des bits de chiffré correspondant à un clair nul. L'état initial correct conduit à un résultat proche de 3/4 ; les autres à une valeur proche de 1/2.

1.3.3 Tests d'hypothèse

Dans l'exemple qui précède, on a fait un certain nombre d'hypothèses successives (sur la configuration initiale du premier LFSR) et on les a testées par un algorithme évaluant la fréquence d'un certain événement. Si cette fréquence est proche de 3/4 — disons supérieure à $\alpha > 1/2$ — on a validé l'hypothèse, sinon on l'a infirmée. Il peut toutefois arriver

1. qu'on rejette une hypothèse correcte (erreur) ;
2. qu'on accepte une hypothèse incorrecte (fausse alerte).

On peut estimer le niveau de confiance c'est à dire la probabilité que l'une ou l'autre situation se produise. On se restreint à la première et on considère que l'événement que l'on observe est la fréquence des succès dans n tirages de Bernoulli indépendants de paramètre $p = 3/4$. On note σ la variance de cette distribution $\sigma = \sqrt{pq}$, où $q = 1 - p$. Le théorème central limite affirme que, si S_n désigne le nombre de succès observés en n tirages, on a :

$$\Pr\left[\frac{S_n - np}{\sigma\sqrt{n}} < \beta\right] \longrightarrow \mathcal{N}(\beta)$$

où $\mathcal{N}(x)$ est la loi normale

$$\mathcal{N}(\beta) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{y^2}{2}} dy$$

On pourra finalement estimer la probabilité d'erreur en fonction du seuil choisi α et de l'échantillon disponible n par $\mathcal{N}(\frac{(\alpha-p)\sqrt{n}}{\sigma})$. Pour que cette probabilité ne soit pas trop grande, il faut que $\sqrt{n}$ excède assez significativement l'inverse de la différence $|\alpha - p|$.

1.4 Cryptographie à clé publique

1.4.1 Éléments de théorie algorithmique des nombres

On rappelle qu'un entier positif n s'écrit de manière unique comme produit de facteurs premiers $n = \prod_{i \in I} p_i^{e_i}$, où I est l'ensemble des indices des nombres premiers qui divisent n. En quotientant $\mathbb{Z}$ par la relation $x = y \bmod n$, qui exprime que $x - y$ est divisible par n, on obtient l'anneau $\mathbb{Z}_n$. On peut choisir de décrire cet anneau en associant à chaque élément x de $\mathbb{Z}$ l'unique représentant $x \bmod n$ de sa classe d'équivalence qui soit positif et strictement plus petit que n. Le théorème des restes chinois énonce que l'application

$$x \in \mathbb{Z}_n \longmapsto (x_i \bmod p_i^{e_i})_{i \in I}$$

est un isomorphisme. Les éléments de $\mathbb{Z}_n$ qui ont un inverse multiplicatif forment un groupe noté $\mathbb{Z}_n^\star$. Ce groupe est isomorphe au produit :

$$\prod_{i \in I} \mathbb{Z}^\star_{p_i^{e_i}}$$

Un élément x de $\mathbb{Z}_n$ est dans $\mathbb{Z}_n^\star$ si et seulement son pgcd avec n est égal à 1. L'algorithme d'Euclide étendu permet alors de calculer les coefficients de Bézout a et b tels que $ax+bn = 1$ et donc d'obtenir l'inverse a de x modulo n. Le nombre d'éléments de $\mathbb{Z}_n$ est noté $\varphi(n)$ et la fonction φ ainsi définie prend le nom d'indicatrice d'Euler. On a alors, pour tout élément x de $\mathbb{Z}_n^\star$:

$$x^{\varphi(n)} = 1 \bmod n$$

L'isomorphisme mis en évidence ci-dessus entraîne l'égalité

$$\varphi(n) = \prod_{i \in I} \varphi(p_i^{e_i})$$

et, en comptant le nombre d'éléments premiers à $p_i^{e_i}$, on obtient $\varphi(p_i^{e_i}) = (p-1)p^{e_i-1}$.

Soit p un nombre premier, alors $\varphi(p) = p - 1$ et donc $\mathbb{Z}_p$ est un corps commutatif. L'égalité $x^{p-1} = 1 \bmod p$ est le "petit" théorème de Fermat. Pour $x \neq 0$, on pose $\binom{x}{p} = x^{\frac{p-1}{2}} \bmod p$. Cette quantité prend le nom de *symbole de Legendre*. Elle vaut 1 ou -1 selon que x est ou non un carré. Quand n n'est pas premier, on peut définir le symbole de Jacobi, noté également $\binom{x}{n}$, en posant

$$\binom{x}{n} = \prod_{i \in I} \binom{x}{p_i}^{e_i}$$

Le symbole de Legendre est facilement calculé en utilisant l'algorithme d'exponentiation modulaire qui prend en entrée trois entiers positifs x, e, n et retourne $x^e \bmod n$ comme suit :

$a = 1$;
pour $i = k - 1$ jusqu'à 0 par pas de -1
    $a = a * a \bmod n$
    si $e_i \neq 0$ $a = a * x \bmod n$
retourner a

Dans le pseudo-code ci-dessus, k est une constante représentant le nombre de bits de e et e_i désigne une fonction qui extrait le i-ème bit de e (comme en C, `e & (1<<i)`). Le symbole de Jacobi est calculé en utilisant d'une part la loi de réciprocité quadratique de Gauss, qui énonce l'identité

$$\binom{m}{n}\binom{n}{m} = (-1)^{\frac{(n-1)(m-1)}{4}}$$

et d'autre part l'égalité

$$\binom{2}{n} = (-1)^{\frac{n^2-1}{8}}$$

1.4.2 RSA

Il est facile de produire des nombres premiers p : une méthode pratique repose sur le test probabiliste de Rabin. Ce test choisit au hasard un entier x, $0 < x < n$, et calcule $x^{n-1} \bmod n$ par la méthode rappelée ci-dessus. On conclut que n n'est pas premier dans les deux cas suivants

1. si le résultat final est $\neq 1$ (le petit théorème de Fermat est contredit) ;

2. si la variable a prend successivement une valeur $\neq \pm 1$ et la valeur 1 (la première de ces valeurs est une racine carrée de l'unité $\neq \pm 1$ et $\mathbb{Z}_n^\star$ n'est pas un corps).

On répète le test de base un nombre prescrit de fois et on déclare premier un nombre pour lequel on n'a pas conclu qu'il ne l'était pas. On démontre que la probabilité d'erreur du test est exponentiellement petite en fonction du nombre de répétitions.

Il est en revanche difficile de calculer la décomposition en facteurs d'un entier. On appelle entier RSA un produit de deux facteurs premiers de même taille $n = pq$. Pour de tels nombres, le record de factorisation est à 512 bits (154 chiffres décimaux), au prix de l'utilisation de plusieurs centaines de machines pendant plusieurs mois.

Soit n un entier RSA et e un entier premier à $\varphi(n)$. Soit d l'inverse de e modulo $\varphi(n)$. On a :

$$(x^e \bmod n)^d \bmod n = (x^d \bmod n)^e \bmod n = x^{ed} \bmod n = x \bmod n$$

On remarque de plus que, si n et e sont donnés, p et q ne sont pas facilement accessibles, ni non plus d. On a donc une situation analogue à celle d'un algorithme de chiffrement et d'un algorithme de déchiffrement conventionnels : la fonction $x \mapsto x^e \bmod n$ joue le rôle de l'algorithme de chiffrement et la fonction $x \mapsto x^d \bmod n$ joue le rôle de l'algorithme de déchiffrement. Toutefois, ces algorithmes prennent en entrée des clés distinctes, e pour le premier, d pour le second et la connaissance de e ne permet pas de déduire d. On a ainsi poussé à l'extrême le principe de Kerckhoffs : même la clé de chiffrement peut passer sans inconvénient aux mains de l'ennemi. Cette possibilité a été découverte par Diffie et Hellman dans [3] et le système RSA a été ensuite proposé par Rivest, Shamir et Adleman dans [7]. Le couple $\mathsf{pk} = (n, e)$ prend le nom de clé publique et permet le chiffrement, l'entier $\mathsf{sk} = d$ est la clé privée, qui autorise le déchiffrement. On note que la connaissance de sk permet de résoudre l'équation

$$X^e = b \bmod n$$

où b est une constante arbitraire, c'est à dire d'extraire des racines e-ièmes arbitraires. L'équation est publique et une solution est publiquement vérifiable. Le RSA permet donc d'offrir le service de non-répudiation, hors d'atteinte de la cryptologie conventionnelle.

1.5 Cryptographie et complexité

1.5.1 Machines de Turing polynomiales

Pour l'évaluation des algorithmes, notre modèle de calcul est la machine de Turing à plusieurs rubans (voir par exemple [9]). La machine peut retourner un bit ou calculer une fonction dont le résultat est sauvegardé sur un ruban particulier. Le temps de calcul est le nombre de pas de calculs avant arrêt de la machine. Sauf exception, ce temps de calcul sera toujours polynomial, c'est à dire borné par un polynôme en fonction de la taille des données. Cette taille des données prend en cryptographie le nom de *paramètre de sécurité* : typiquement, c'est la taille de l'entier RSA n de la section précédente.

Un prédicat polynomial est une relation $R(x, y)$ qui peut être testée par une machine de Turing polynomiale (en fonction du paramètre de sécurité). On requiert de plus que les tailles de x et y restent polynomiales (toujours en fonction du paramètre de sécurité). Un problème de la classe NP consiste, sur une donnée x, en la recherche d'un élément y tel que $R(x, y)$, où R est un prédicat polynomial.

Il existe pas mal de variantes de la machine de Turing, dont la cryptologie fait usage — au moins dans sa partie la plus théorique. Une machine de Turing polynomiale probabiliste

dispose d'un ruban spécifique dit ruban d'aléa, contenant une suite de bits Ω de taille assez longue pour qu'on n'en manque pas. C'est donc une machine normale, à cela près que les différentes configurations initiales du ruban d'aléa peuvent s'interpréter comme un espace de probabilité, muni de la distribution uniforme. Les configurations successives de la machine deviennent des variables aléatoires, de même par exemple que le résultat du calcul. Si la machine retourne un bit, on peut ainsi calculer la probabilité que ce bit soit à un.

Une machine à oracle permet le recours à un sous programme sur lequel on ne fait pas d'hypothèse de complexité. Elle dispose d'un ruban particulier, dit ruban d'oracle. Lorsque son programme l'indique, elle peut soumettre le contenu du ruban d'oracle et l'oracle retourne la fonction qu'il a la charge de calculer (en un pas de calcul). Cette notion permet en particulier de comparer les problèmes algorithmiques : une réduction polynomiale d'un problème à un autre est une machine polynomiale à oracle qui résout le premier problème à l'aide d'un oracle pour le second. Un problème est NP-complet si tout autre problème NP s'y réduit. Les *cognoscenti* feront remarquer que ce n'est pas là la définition classique et qu'on a substitué la réduction de Cook à celle de Karp : pour cette dernière, il y a un unique appel à l'oracle qui fournit le résultat final du calcul.

Un type particulier d'oracle est l'oracle aléatoire (voir [2]) : il retourne sur chaque question une réponse aléatoire. On requiert seulement qu'il soit consistant, en ce sens qu'il doit donner des réponses identiques à des questions égales. L'oracle aléatoire est un modèle imparfait de fonctions dont le comportement est générique, par exemple les fonctions de hachage. Mathématiquement, il n'y a plus d'oracle : on peut simplement considérer une pile polynomiale des réponses successives aux différentes questions, munie d'une distribution uniforme, comme un ruban d'aléa.

Enfin, une machine de Turing polynomiale probabiliste peut être munie de rubans d'interaction. Ces rubans permettent de modéliser les communications entre machines. Les messages reçus d'une autre machine sont placés dans un ruban de réception, ceux à destination d'une machine distante sont sauvegardés dans un ruban d'envoi. On note MTTPI le modèle de machine interactive obtenu.

1.5.2 Réduction et simulation

Dans l'approche algorithmique de la cryptologie, les algorithmes cryptographiques sont exécutés par des MTPPI. Les adversaires sont également des MTTPI et les divers scénarios de cryptanalyse précisent les ressources dont ils disposent. Par exemple une attaque "à chiffré choisi" autorise l'adversaire à interagir avec une machine qui exécute l'algorithme de déchiffrement. La méthode fondamentale est de prouver l'existence d'une réduction : en utilisant l'adversaire comme oracle, on parvient à résoudre un problème déterminé réputé difficile, comme la factorisation des entiers. En l'absence de réponse à la question ouverte $P = NP$, on ne peut de toutes façons pas faire mieux. Une tendance récente vise toutefois à quantifier la sécurité, c'est à dire à proposer des réductions "efficaces" : le temps de calcul de la réduction et sa probabilité de succès doivent alors être optimisés.

La cryptologie "théorique" modélise également la protection du secret : l'exécution des algorithmes cryptographiques par des MTTPI produit une trace visible pour l'adversaire : ce dernier peut en effet

– intercepter les communications : on parle d'adversaire *passif*;

– interagir avec les machines exécutant les algorithmes : on parle d'adversaire *actif*.

L'information recueillie est la *vue* de l'adversaire, qui peut ensuite mettre en œuvre les méthodes statistiques, telles que celles de la section 1.3. Bien entendu la vue de l'adversaire

dépend de divers secrets, comme par exemple une clé privée sk d'un algorithme RSA. Pour garantir que sk reste à l'abri des attaques statistiques, on utilise la notion de simulation, qui est en somme la version algorithmique de la sécurité à la Shannon. Un simulateur est une MTTP qui produit une vue simulée, sans accéder à sk. On souhaite montrer que les deux vues sont essentiellement identiques ; pour cela on les considère comme des distributions de probabilité sur l'ensemble des vues possibles.

1. si ces distributions sont identiques, on parle de simulation parfaite ;
2. si ces distribution ont une distance négligeable on parle de simulation statistique ;
3. si aucun test probabiliste ne distingue les deux distributions de façon non négligeable, on parle de simulation algorithmique.

Dans ce qui précède le mot négligeable peut faire référence à des évaluations concrètes ou à une estimation asymptotique : est négligeable une quantité qui décroît exponentiellement avec le paramètre de sécurité, ou plus vite que l'inverse de tout polynôme. Un test statistique retourne zéro ou 1. Sa probabilité de succès est la probabilité d'obtenir 1. On distingue deux distributions en calculant la différence des deux probabilités de succès.

Bibliographie

[1] *AES Advanced Encryption Standard*, `http://csrc.nist.gov/encryption/aes/`

[2] M. Bellare & P. Rogaway. Random Oracles Are Practical : a Paradigm for Designing Efficient Protocols. In *Proc. of the 1st CCS*, pages 62–73. ACM Press, New York, 1993.

[3] W. Diffie & M. E. Hellman. New Directions in Cryptography, *IEEE Trans. Inform. Theory*, IT-22, 1976, 644–654.

[4] A. Kerckhoffs. *La cryptographie militaire*, Paris, 1883.

[5] LOI no 2000-230 du 13 mars 2000 portant adaptation du droit de la preuve aux technologies de l'information et relative à la signature électronique `http://www.adminet.com/jo/20000314/`

[6] A.J. Menezes, P.C. van Oorschot & S.A. Vanstone. *Handbook of applied cryptography*, CRC Press, new York, 1997.

[7] R. Rivest, A. Shamir & L. Adleman. A Method for Obtaining Digital Signatures and Public Key Cryptosystems. *Communications of the ACM*, 21(2) :120–126, February 1978.

[8] B. Schneier, *Applied Cryptography*, 2nd ed., John Wiley & Sons, New York, (1995).

[9] J. Stern. *Fondements mathématiques de l'informatique*, Mac Graw Hill, Paris (1990).

[10] J. Stern. *La science du secret*, Éditions Odile Jacob, Paris, (1998).

[11] D. Stinson. *Cryptographie, théorie et pratique*, Thomson Publishing France, Paris, 1996.

[12] Tripwire Inc. *Tripwire, 2001*, `http://www.tripwire.com/`

Chapitre 2 (6h)

Chiffrement par bloc et cryptanalyse différentielle

— par Louis Granboulan

Sommaire

2.1 Modes d'opération 18
2.1.1 Modes d'opération d'un système de chiffrement par bloc 18
2.1.2 Modes classiques 19
2.1.3 Modes de flot 21
2.1.4 Intégrité 24
2.2 Principes de conception 25
2.2.1 Modèle de sécurité 25
2.2.2 Chiffrement itératif et diversification de clef 26
2.2.3 Confusion et diffusion 26
2.2.4 Quelques exemples 26
2.2.5 Réseaux à base de boîtes de substitution 27
2.2.6 Réseaux de type Feistel 28
2.2.7 Autres constructions 28
2.3 Étude théorique des schémas de Feistel 29
2.3.1 Attaques utilisant un distingueur 29
2.3.2 Attaques génériques 29
2.3.3 Preuves de sécurité 30
2.4 Cryptanalyse différentielle 30
2.4.1 Description de l'attaque 30
2.4.2 Mise en œuvre 32
2.4.3 Résistance prouvée 34
2.5 Variantes de la cryptanalyse différentielle 34
2.5.1 Différentielles tronquées 35
2.5.2 Différentielles impossibles 35
2.5.3 Différentielles d'ordre supérieur 35
2.5.4 Boomerang 35
2.6 Autres techniques de cryptanalyse 36
2.6.1 Cryptanalyse linéaire 36

2.6.2 Attaque par interpolation 36
2.6.3 Attaque par résolutions de système d'équations algébriques . . . 36
2.6.4 Attaque par décalage 36
2.6.5 Attaque par saturation 37

Ce chapitre se décompose en deux ensembles, de trois sections chacun. Les trois premières sections présentent les grands principes de conception des systèmes de chiffrement par bloc, sans réellement étudier leur sécurité pratique. Les suivantes présentent les techniques de cryptanalyse qui permettent d'en évaluer la sécurité, en particulier la cryptanalyse différentielle, qui illustre les difficultés d'analyse des cryptosystèmes symétriques.

2.1 Modes d'opération

2.1.1 Modes d'opération d'un système de chiffrement par bloc

L'objectif est d'avoir un service de confidentialité Alice veut transmettre à Bob un ou plusieurs messages, par un canal peu sûr. Alice et Bob se sont auparavant mis d'accord sur une convention publique (choix d'un système de chiffrement par bloc et d'un mode d'opération), et sur une donnée secrète (la clef). Un adversaire (Ève) peut écouter les communications (attaques passives), et éventuellement les modifier (attaques actives).

Définition 2.1 (Système de chiffrement par bloc) *Les entiers b et k sont respectivement la taille (en bits) d'un bloc et celle de la clef. Un système de chiffrement par bloc est une paire de fonctions E et D associant à toute clef k de $\mathcal{K} = \{0,1\}^k$ une permutation $\mathsf{E_k}$ de l'espace des blocs $\{0,1\}^b$, et la permutation inverse $\mathsf{D_k}$.*

Définition 2.2 (Mode d'opération) *On appelle mode d'opération d'un système de chiffrement par bloc une technique qui permet, avec une unique clef k, de chiffrer ou déchiffrer un message dont la longueur n'est pas nécessairement b, au moyen d'un certain nombre d'appels de la fonction $\mathsf{E_k}$ ou $\mathsf{D_k}$.*

Certains systèmes de chiffrement par bloc autorisent plusieurs tailles de bloc, mais ce n'est pas la même chose que l'utilisation d'un mode de chiffrement. En particulier, le temps de chiffrement de tels systèmes subit un accroissement non linéaire en la taille du bloc. De plus, les contraintes de sécurité et leur analyse dépendent fortement de b. Il ne faut donc pas considérer que cette souplesse dans le choix de la taille du bloc est une souplesse pour la taille des messages chiffrés, mais plutôt un choix de sécurité.

Toute implantation d'un système de chiffrement par bloc doit donc choisir un mode d'opération. Quatre modes (ECB, CBC, CFB et OFB) ont été normalisés en 1980 par le NIST (anciennement NBS : National Bureau of Standards, organisme américain de normalisation). De nombreux autres modes ont été proposés dans divers contextes, et le NIST a commencé en 2001 une nouvelle sélection [31].

Propriétés. Bien évidemment, tout mode d'opération doit permettre de déchiffrer. La description d'un mode d'opération explique donc comment partir d'un message m de longueur quelconque pour obtenir un chiffré c, et comment à partir de ce chiffré c obtenir le message m. La seule information dont le secret doit garantir la confidentialité de m, connaissant c, est la clef k.

Un mode d'opération est *sans expansion* si le nombre de bits du chiffré c est exactement égal au nombre de bits du message m. Le mode d'opération peut éventuellement dépendre d'une *valeur d'initialisation* publique et convenue à l'avance entre Alice et Bob. Cette valeur doit pouvoir être choisie par l'attaquant sans que cela n'affaiblisse le système. [1]

L'opération de déchiffrement est déterministe : son résultat est le message clair, ou bien une erreur si le message chiffré ne correspond à aucun clair. Un mode d'opération *garantit l'intégrité* si une modification de c par l'attaquant est détectée au moment de déchiffrer. C'est une protection nécessaire contre les attaques actives.

Remarquons qu'un mode d'opération garantissant l'intégrité avec probabilité d'erreur au plus 2^{-l} a une expansion d'au moins l bits. En effet, pour tout mode d'opération ayant une expansion de l bits, un attaquant peut remplacer le chiffré par une valeur aléatoire, qui est un chiffré valide avec probabilité au moins 2^{-l}.

Réinitialisation. Dans un grand nombre d'applications, on veut pouvoir chiffrer plusieurs messages avec la même clef secrète, de manière asynchrone. On utilise alors souvent une valeur IV (Initialisation Value), publique, différente pour chaque message. Lorsqu'on veut être protégé contre les attaques actives, on considère que l'adversaire peut choisir la valeur IV utilisée.

2.1.2 Modes classiques

Mode ECB

Définition. ECB signifie Electronic CodeBook. Le message est découpé en blocs de taille b. Chaque bloc est chiffré séparément par $\mathsf{E_k}$, le chiffré est la concaténation des blocs obtenus.

Le déchiffrement se fait de façon similaire. Le message est découpé en blocs de taille b. Chaque bloc est déchiffré séparément par $\mathsf{D_k}$, le clair est la concaténation des blocs obtenus.

Chiffrement et déchiffrement en mode ECB.

Propriétés.

- **Expansion.**
  C'est un mode sans expansion, mais qui ne marche qu'avec des messages dont la longueur est un multiple de b. On peut généraliser le mode ECB aux autres longueurs de message, par exemple en complétant le dernier bloc avec un bit à 1 suivi du nombre adéquat de bits à 0. Le mode ECB a alors une expansion de 1 à b bits.
  La technique appelée *Ciphertext stealing* [12] permet de chiffrer sans expansion les messages de longueur au moins b. Appelons m_{n-1} et m_n les deux derniers blocs, le

[1] Ceci est différent d'un système *randomisé*, pour lequel l'opération de chiffrement utilise une valeur aléatoire qui n'est pas transmise en clair. Un même message donne ainsi plusieurs chiffrés distincts. Il y a alors nécessairement expansion du message. En pratique, seuls les systèmes de chiffrement asymétriques sont randomisés.

dernier pouvant être incomplet ; leur chiffrement donne les deux blocs c_{n-1} et c_n de mêmes tailles, tels qu'il existe une valeur c' avec $\mathsf{E_k}(m_{n-1}) = c_n||c'$ et $\mathsf{D_k}(c_{n-1}) = m_n||c'$.

– **Performance.**
Le mode ECB peut être totalement parallélisé : le chiffrement d'un bloc ne dépend pas du chiffrement des autres.

– **Résistance aux erreurs de transmission.**
Si un bloc c_i est modifié, seul le bloc m_i correspondant sera modifié.
Si un nombre de bits multiple de b est perdu par la transmission, seuls les blocs correspondants sont perdus.

Sécurité. Le mode ECB ne présente aucune détection d'intégrité et est donc vulnérable aux attaques actives. En particulier si le format du texte clair est connu : l'ordre des blocs à l'intérieur peut être changé, et si plusieurs messages ont été chiffrés avec la même clef, ils peuvent être facilement mélangés par l'attaquant.

De plus, des blocs identiques dans le message clair sont transformés en des blocs identiques dans le message chiffré, ce qui donne facilement de l'information à un attaquant passif.

Le mode ECB est le plus simple qu'on puisse imaginer, mais c'est aussi celui dont la sécurité est la plus faible. Sauf pour des applications très particulières, le mode ECB est à éviter.

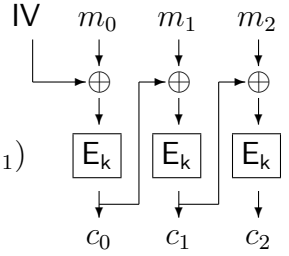
Mode CBC

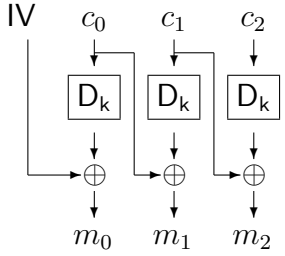
Définition. CBC signifie Cipher Block Chaining. Alice et Bob ont au préalable convenu d'une valeur publique IV faisant b bits. Le message est découpé en blocs de taille b. Le chiffrement d'un bloc se calcule en chiffrant par $\mathsf{E_k}$ le ou bit-à-bit (XOR) du bloc clair et du bloc chiffré précédent. La valeur IV sert de chiffré précédant le bloc 0.

Le déchiffrement se fait de façon similaire. Un bloc du clair est obtenu en déchiffrant avec $\mathsf{D_k}$ puis en faisant un ou bit-à-bit avec le bloc chiffré précédent.

$c_{-1} = \mathsf{IV}$

$c_i = \mathsf{E_k}(m_i \oplus c_{i-1})$





$m_i = \mathsf{D_k}(c_i) \oplus c_{i-1}$

Chiffrement et déchiffrement en mode CBC.

Propriétés.

– **Expansion.**
Si la valeur de IV a été convenue à l'avance, c'est un mode sans expansion, mais qui ne marche qu'avec des messages dont la longueur est un multiple de b. On peut généraliser le mode CBC aux autres longueurs de message, de la même façon que le mode ECB.

– **Performance.**
Le chiffrement CBC ne peut être parallélisé, mais le déchiffrement CBC peut être totalement parallélisé.

– **Résistance aux erreurs de transmission.**
Si un bloc c_i est modifié, seul les blocs m_i et m_{i+1} correspondants seront modifiés.
Si un nombre de bits multiple de b est perdu par la transmission, seuls les blocs correspondants sont perdus.

Sécurité. Le mode CBC ne présente aucune détection d'intégrité et est donc vulnérable aux attaques actives. En revanche, il a une bonne sécurité face aux attaques passives. On peut déduire de l'information sur le message dès que le chiffré contient deux blocs égaux : si $c_i = c_j$, alors $m_i \oplus m_j = c_{i-1} \oplus c_{j-1}$. Si les c_i peuvent être considérés comme aléatoires, alors le paradoxe des anniversaires affirme qu'une collision apparaît lorsque $2^{b/2}$ blocs ont été chiffrés.

Il n'y a pas d'objection à chiffrer plusieurs messages avec la même clef, avec une valeur IV aléatoire différente à chaque fois. En revanche, si l'attaquant a un contrôle sur IV, il ne faut pas chiffrer plusieurs messages avec la même clef. On conseille donc d'utiliser $\mathsf{E_k}(\mathsf{IV})$ au lieu de IV, mais alors le déchiffrement utilise $\mathsf{E_k}$ et $\mathsf{D_k}$.

Le mode CBC est le mode utilisé dans la plupart des applications des systèmes de chiffrement par bloc.

2.1.3 Modes de flot

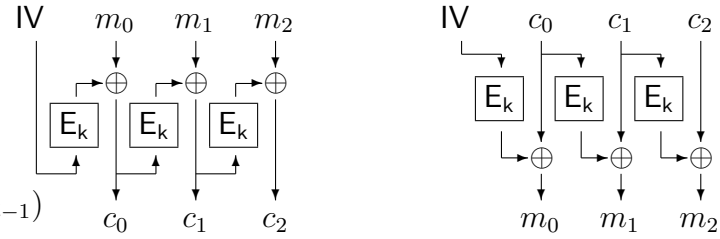
Mode CFB

Définition. Le mode CFB (Cipher FeedBack) fabrique un registre à décalage en mode CTAK (CipherText Auto Key), pour obtenir un chiffrement de flot.

Le mode CFB est paramétré par un entier ℓ inférieur à b et utilise un registre de b bits, initialisé par une valeur publique IV (Initial Value) convenue à l'avance. Le message est découpé en blocs de taille ℓ. À chaque coup d'horloge, on calcule l'image du registre par $\mathsf{E_k}$, dont on extrait ℓ bits qu'on appelle r_i. Le bloc est chiffré par ou exclusif : $c_i = m_i \oplus r_i$. La nouvelle valeur du registre est obtenue en faisant un décalage de ℓ bits, et en y entrant la valeur c_i.

Chiffrement	$s_{-1} = \mathsf{IV}$	$r_i = [\mathsf{E_k}(s_{i-1})]_{(1\ldots\ell)}$	$c_i = m_i \oplus r_i$	$s_i = [s_{i-1}]_{(\ell+1\ldots b)} \| c_i$
Déchiffrement	$s_{-1} = \mathsf{IV}$	$r_i = [\mathsf{E_k}(s_{i-1})]_{(1\ldots\ell)}$	$m_i = c_i \oplus r_i$	$s_i = [s_{i-1}]_{(\ell+1\ldots b)} \| c_i$

$c_{-1} = \mathsf{IV}$
$c_i = m_i \oplus \mathsf{E_k}(c_{i-1})$



$m_i = c_i \oplus \mathsf{E_k}(c_{i-1})$

Chiffrement et déchiffrement en mode CFB, avec $\ell = b$.

Propriétés.

– **Expansion.**
Si la valeur de IV a été convenue à l'avance, c'est un mode sans expansion : si la longueur du message n'est pas un multiple de ℓ, on utilise une valeur tronquée pour le dernier r_i.

– **Performance.**
Le chiffrement CFB ne peut être parallélisé, mais le déchiffrement CFB peut être totalement parallélisé.
Le chiffrement et le déchiffrement utilisent tous deux la fonction $\mathsf{E_k}$, et donc le temps de calcul de $\mathsf{D_k}$ n'a aucune influence sur les performances du mode CFB. En revanche, il faut un appel à $\mathsf{E_k}$ tous les ℓ bits du message.
– **Résistance aux erreurs de transmission.**
Si un bloc c_i est modifié, seul le bloc m_i et quelques autres seront modifiés.
Si un nombre de bits multiple de ℓ est perdu par la transmission, seuls les blocs correspondants sont perdus.

Sécurité. Le mode CFB ne présente aucune détection d'intégrité et est donc vulnérable aux attaques actives. Comme le mode CBC, c'est à partir du chiffrement de $2^{b/2}$ blocs qu'un attaquant obtient éventuellement de l'information sur le message (grâce au paradoxe des anniversaires).

Les contraintes de sécurité sur la fonction $\mathsf{E_k}$ sont d'autant moins sévères que ℓ est petit.

Il n'y a pas d'objection à chiffrer plusieurs messages avec la même clef, avec une valeur IV différente à chaque fois, si cette valeur est choisie aléatoirement. On conseille donc dans ce contexte l'utilisation de $\mathsf{E_k}(\mathsf{IV})$ au lieu de IV.

Mode OFB

Définition. Le mode OFB (Output FeedBack) fabrique un registre à décalage en mode KAK (Key Auto Key), pour obtenir un chiffrement de flot.

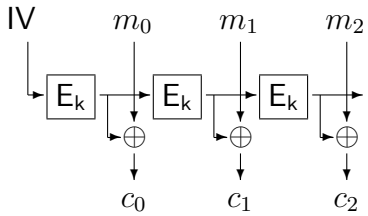
Le mode OFB est paramétré par un entier ℓ inférieur à b et utilise un registre de b bits, initialisé par une valeur publique IV (Initial Value) convenue à l'avance. Le message est découpé en blocs de taille ℓ. À chaque coup d'horloge, on calcule l'image du registre par $\mathsf{E_k}$, dont on extrait ℓ bits qu'on appelle r_i. Le bloc est chiffré par ou exclusif : $c_i = m_i \oplus r_i$. La nouvelle valeur du registre est obtenue en faisant un décalage de ℓ bits, et en y entrant la valeur r_i.

Chiffrement	$s_{-1} = \mathsf{IV}$	$r_i = [\mathsf{E_k}(s_{i-1})]_{(1\ldots\ell)}$	$c_i = m_i \oplus r_i$	$s_i = [s_{i-1}]_{(\ell+1\ldots b)} \| r_i$
Déchiffrement	$s_{-1} = \mathsf{IV}$	$r_i = [\mathsf{E_k}(s_{i-1})]_{(1\ldots\ell)}$	$m_i = c_i \oplus r_i$	$s_i = [s_{i-1}]_{(\ell+1\ldots b)} \| r_i$

$s_{-1} = \mathsf{IV} \quad s_i = \mathsf{E_k}(s_{i-1})$

$c_i = m_i \oplus s_i$

$m_i = c_i \oplus s_i$



Chiffrement ou déchiffrement en mode OFB, avec $\ell = b$.

Propriétés.
– **Expansion.**
Si la valeur de IV a été convenue à l'avance, c'est un mode sans expansion : si la longueur du message n'est pas un multiple de ℓ, on utilise une valeur tronquée pour le dernier r_i.

– **Performance.**
Le chiffrement ni le déchiffrement OFB ne peuvent être parallélisés.
La séquence des s_i peut être précalculée avant de connaître le message.
Le chiffrement et le déchiffrement utilisent tous deux la fonction $\mathsf{E_k}$, et donc le temps de calcul de $\mathsf{D_k}$ n'a aucune influence sur les performances du mode OFB. En revanche, il faut un appel à $\mathsf{E_k}$ tous les ℓ bits du message.
– **Résistance aux erreurs de transmission.**
Si un bloc c_i est modifié, seul le bloc m_i sera modifié.
Si quelques bits sont perdus par la transmission, toute la fin du message est perdue.

Sécurité. Le mode OFB ne présente aucune détection d'intégrité et est donc vulnérable aux attaques actives.

Les contraintes de sécurité sur la fonction $\mathsf{E_k}$ sont d'autant moins sévères que ℓ est petit.

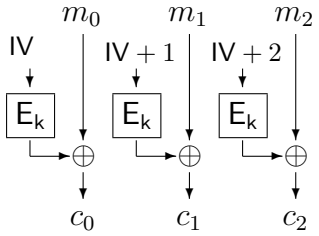
Pour $\ell = b$, si on utilise une fonction $\mathsf{E_k}$ indistinguable d'une permutation aléatoire, alors la valeur s_i fera un cycle avant d'avoir parcouru les 2^b valeurs possibles. On pourra donc préférer utiliser un système de chiffrement par blocs tel que chaque permutation $\mathsf{E_k}$ a un unique cycle. [2]

Il n'y a pas d'objection à chiffrer plusieurs messages avec la même clef, avec une valeur IV différente à chaque fois, si cette valeur est choisie aléatoirement. On conseille donc dans ce contexte l'utilisation de $\mathsf{E_k}(\mathsf{IV})$ au lieu de IV.

Mode CTR

Définition. CTR signifie CounTeR. La valeur IV est convenue à l'avance, et on fabrique le flot $\mathsf{E_k}(\mathsf{IV}), \mathsf{E_k}(\mathsf{IV}+1), \mathsf{E_k}(\mathsf{IV}+2), ...$ qui sert à chiffrer le message par ou exclusif.

$s_i = \mathsf{E_k}(\mathsf{IV} + i)$
$c_i = m_i \oplus s_i$
$m_i = c_i \oplus s_i$



Chiffrement ou déchiffrement en mode CTR.

Propriétés.
– **Expansion.**
Si la valeur de IV a été convenue à l'avance, c'est un mode sans expansion : si la longueur du message n'est pas un multiple de b, on utilise une valeur tronquée pour le dernier s_i.
– **Performance.**
Le chiffrement et le déchiffrement CTR peuvent être parallélisés.
La séquence des s_i peut être précalculée avant de connaître le message.
Le chiffrement et le déchiffrement utilisent tous deux la fonction $\mathsf{E_k}$, et donc le temps de calcul de $\mathsf{D_k}$ n'a aucune influence sur les performances du mode CTR.

[2] En moyenne, la longueur du plus long cycle d'une permutation aléatoire de 2^b valeurs est environ $0.6\ 2^b$, mais une telle permutation a très probablement un petit cycle, dans lequel la valeur IV peut avoir été choisie. Si la fonction $\mathsf{E_k}$ n'est pas une permutation, elle peut aussi être utilisée en mode OFB, mais alors le cycle a très probablement longueur $2^{b/2}$.

– **Résistance aux erreurs de transmission.**
Si un bloc c_i est modifié, seul le bloc m_i sera modifié.
Si quelques bits sont perdus par la transmission, toute la fin du message est perdue.

Sécurité. Le mode CTR ne présente aucune détection d'intégrité et est donc vulnérable aux attaques actives.

Comme la fonction E_k est une permutation, le flot produit ne cycle qu'après 2^b blocs.

Il n'y a pas d'objection à chiffrer plusieurs messages avec la même clef, avec une valeur IV différente à chaque fois, si cette valeur est choisie aléatoirement. On conseille donc dans ce contexte l'utilisation de $\mathsf{E}_\mathsf{k}(\mathsf{IV})$ au lieu de IV.

2.1.4 Intégrité

Chiffrement + MAC

Pour garantir l'intégrité du message, on y ajoute un suffixe (*tag*) calculé par un MAC (Message Authentication Code). Le résultat calculé par un MAC dépend du message et d'une clef secrète, de telle sorte qu'il soit en pratique nécessaire de connaître la clef pour vérifier le tag. Si la clef k' du MAC est choisie indépendamment de la clef k du système de chiffrement, alors la sécurité est optimale. C'est ce que Shoup [40] appelle DEM1 (Data Encapsulation Mechanism 1).

L'exemple classique d'algorithme de MAC est le CBC-MAC (décrit par exemple parmi les modes du DES [30]), pour lequel le tag s'obtient en chiffrant le message en mode CBC et en gardant uniquement le dernier bloc chiffré. Souvent, la combinaison confidentialité+intégrité est obtenue à l'aide d'un chiffrement CBC, plus un CBC-MAC, tous les deux avec la même clef k, mais des valeurs IV distinctes. Par exemple l'une est obtenue par image de l'autre par E_k.

Il existe aussi des algorithmes spécifiques calculant un MAC plus rapidement qu'un chiffrement. Par exemple HMAC [32], UMAC [10], ...

Chiffrement + Checksum

Le coût du calcul d'un MAC n'étant pas négligeable, diverses techniques ont été proposées pour obtenir un chiffrement garantissant l'intégrité avec un surcoût très faible par rapport à un simple chiffrement.

L'idée la plus naturelle est de rajouter un bloc de checksum (non cryptographique) à la fin du message, et de chiffrer l'ensemble. Le checksum le plus rapide à calculer est un XOR des blocs du messages : c'est ainsi que marche le mode appelé CBCC.

Presque tous les modes basés sur cette idée ont des faiblesses. Par exemple on attaque facilement le mode CBCC dans le cas du chiffrement d'un message d'un unique bloc. Le mode ECB + Checksum est pire, puisqu'il n'offre aucune protection contre le changement de l'ordre des blocs du message...

Mode OCB

Le mode OCB a été proposé par Rogaway [38]. C'est une variante du mode IAPM proposé par Jutla [17], lui-même l'un des premiers modes de chiffrement dont la sécurité est prouvée et qui assure l'intégrité pour un surcoût faible.

Le mode OCB est une variante du mode ECB+Checksum, où l'on rajoute avant et après chaque chiffrement une opération simple[3] dépendant de la clef et protégeant des attaques actives. De plus, le chiffrement du dernier bloc est différent (et ressemble au mode CTR) afin d'avoir un mode avec expansion minimale. Les détails du mode OCB ont été choisis pour en optimiser les performances et la facilité d'implantation.

2.2 Principes de conception

2.2.1 Modèle de sécurité

Oracles. Un système de chiffrement par bloc doit résister aux attaques passives, à clair et à chiffré choisis, adaptatives. En pratique, on se place dans le scénario ci-après.

La clef secrète k étant fixée et inconnue de l'attaquant, on lui fournit un bloc chiffré c. L'attaquant doit trouver $\mathsf{D}_\mathsf{k}(c)$. On lui donne accès à un oracle de chiffrement (il peut calculer $\mathsf{E}_\mathsf{k}(m_i)$ pour n'importe quelle valeur m_i) et à un oracle de déchiffrement (il peut calculer $\mathsf{D}_\mathsf{k}(c_i)$ pour n'importe quelle valeur c_i distincte de c).

Attaques génériques. Il existe deux attaques génériques : la recherche exhaustive de la clef détecte pour toutes les clefs possibles laquelle est compatible avec l'oracle de (dé)chiffrement ; la recherche exhaustive du message appelle l'oracle de chiffrement pour toutes les valeurs possibles, pour détecter laquelle correspond au chiffré c. L'attaque par recherche exhaustive de la clef demande 2^k calculs et quelques appels à l'oracle. L'attaque par recherche exhaustive du message demande 2^b appels à l'oracle.

On considérera qu'un système de chiffrement par bloc est cassé s'il existe une attaque plus efficace que les attaques génériques, c'est-à-dire utilisant moins de 2^k calculs, moins de 2^b appels à un oracle, et moins de 2^k en espace mémoire.

Clefs liées. De plus, on veut aussi que le système de chiffrement par bloc résiste aux attaques à clefs liées. On suppose que l'attaquant peut avoir accès à un oracle de chiffrement $\mathsf{E}_{f(\mathsf{k})}$ et de déchiffrement $\mathsf{D}_{f(\mathsf{k})}$ pour une clef $f(\mathsf{k})$ liée à la clef k par une fonction f au choix de l'attaquant. [4]

On veut aussi qu'il soit impossible de trouver une paire de clefs k, k' et un message m tels que $\mathsf{E}_\mathsf{k}(m) = \mathsf{E}_{\mathsf{k}'}(m)$. Ceci permet de construire la fonction de hachage sans collisions $\mathsf{k} \mapsto \mathsf{E}_\mathsf{k}(0)$.

Clefs faibles. Ce sont les clefs dont on détecte l'utilisation plus rapidement qu'une clef aléatoire. Par exemple DES a quelques valeurs de k pour lesquelles $\mathsf{E}_\mathsf{k} = \mathsf{D}_\mathsf{k}$. L'existence de clefs faibles permet dans certaines circonstances d'attaquer le cryptosystème. On choisit donc d'exclure les éventuelles clefs faibles de l'espace des clefs $\mathcal{K}$.

Autres propriétés. Enfin, on peut rajouter quelques critères de sécurité particuliers à certaines applications. Par exemple, si le système de chiffrement est destiné à être utilisé en mode OFB, il est important que la permutation E_k n'ait pas de petit cycle.

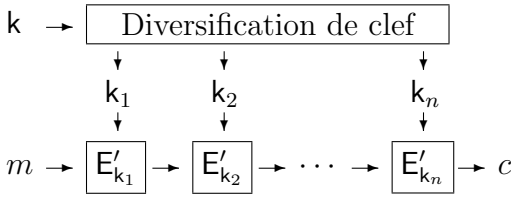
[3] Un ou exclusif avec une valeur calculée à partir d'un code (correcteur) de Gray.

[4] En pratique, on limite le choix de la fonction f à des fonctions très simples telles l'addition ou le ou exclusif d'une constante.

2.2.2 Chiffrement itératif et diversification de clef

Définition 2.3 (Chiffrement itératif) *Pour construire un cryptosystème sûr, on répète plusieurs fois un cryptosystème de moindre sécurité, appelé tour* (round)*. On construit ainsi un cryptosystème à n tours :* $\mathsf{E}_{\mathsf{k}_1 \ldots \mathsf{k}_n} = \mathsf{E}'_{\mathsf{k}_n} \circ \ldots \circ \mathsf{E}'_{\mathsf{k}_1}$.

Définition 2.4 (Diversification de clef) *Les systèmes de chiffrement par bloc incorporent un algorithme de diversification de clef* (key schedule) *qui transforme une clef courte* k *en la clef longue* $\mathsf{k}' = \mathsf{k}_1 \ldots \mathsf{k}_n$*, utilisée au cours du chiffrement.*



L'algorithme de diversification de clef doit être calibré pour que la longueur de la clef k donne une mesure de la sécurité du système.

En pratique, on commence par rechercher la meilleure attaque contre le système où les clefs k_i sont indépendantes. Puis on utilise les particularités de la diversification de clef pour améliorer cette attaque.

Tous les systèmes modernes de chiffrement par bloc sont des systèmes de chiffrement itératifs avec une diversification de clef. Souvent, les premiers ou derniers tours peuvent être différents des autres, et éventuellement plusieurs types de tours peuvent alterner.

2.2.3 Confusion et diffusion

Les deux propriétés nécessaires pour qu'un système de chiffrement ait une bonne sécurité ont été nommées *confusion* et *diffusion* par Shannon en 1949.

La confusion sert à cacher la relation entre le clair et le chiffré. Elle s'obtient habituellement par des opérations (non linéaires) de substitution de sous-blocs.

La diffusion sert à cacher la redondance dans le message et à diffuser sur tout le chiffrer l'influence du changement d'un bit de clef ou d'un bit du clair. Elle s'obtient habituellement par une opération linéaire sur tout le bloc.

La sous-clef intervient habituellement par ou exclusif.

2.2.4 Quelques exemples

La difficulté de la conception d'un algorithme de chiffrement est d'obtenir une sécurité suffisante tout en ayant de bonnes performances. Il faut donc minimiser le nombre de tours, chacun ayant une bonne confusion et une bonne diffusion, tout en étant facile à calculer.

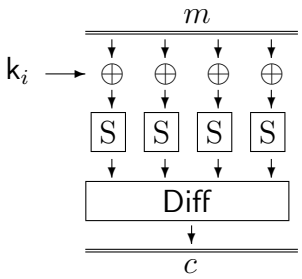
Les algorithmes de chiffrement par bloc sont donc tous le résultat d'un compromis entre la sécurité et les performances, dépendant de l'architecture choisie. Quelques exemples de systèmes de chiffrement par bloc sont :

- DES (Data Encryption Standard) qui a été la norme américaine [27] depuis 1974 et est très largement utilisé dans les applications commerciales. L'algorithme DEA (Data Encryption Algorithm) chiffre des blocs de 64 bits à l'aide d'une clef de 56 bits. Cet algorithme est conçu plus particulièrement pour des implantations matérielles (hardware dédié).

Triple-DES améliore la sécurité de DES en gardant une option de compatibilité ascendante. Sa clef fait 112 bits. Le chiffrement se fait par $\mathrm{DES}_{\mathsf{k}} \circ \mathrm{DES}^{-1}_{\mathsf{k}'} \circ \mathrm{DES}_{\mathsf{k}}$.
- IDEA a été développé entre 1990 et 1992 par Lai et Massey. Cet algorithme chiffre des blocs de 64 bits avec une clef de 128 bits. Il est conçu plus particulièrement pour des microprocesseurs 16 ou 32 bits. Sa principale utilisation est dans le système PGP.
- Skipjack [28] a été développé en 1990 par la NSA et a été rendu public en 1998. Il chiffre des blocs de 64 bits avec une clef de 80 bits, et est conçu plus particulièrement pour des microprocesseurs 8 bits.
- AES [29] est le résultat d'une compétition organisée à partir de 1997 par le NIST pour trouver un successeur au DES. Il chiffre des blocs de 128 bits et accepte des clefs de 128, 192 ou 256 bits. C'est l'algorithme Rijndael [14], développé par les Belges Daemen et Rijmen, qui a été sélectionné. Il est conçu pour être performant quelque soit l'architecture, avec une préférence pour les microprocesseurs 32 bits ou 8 bits.

2.2.5 Réseaux à base de boîtes de substitution

Principe de construction. Chaque tour est la succession de trois opérations : $\mathsf{E}'_{\mathsf{k}_i}(m) = \mathsf{Diff} \circ \mathsf{Subs} \circ \mathsf{Add}_{\mathsf{k}_i}(m)$.
- $\mathsf{Add}_{\mathsf{k}_i}$ ajoute par ou exclusif une sous-clef de b bits.
- Subs applique en parallèle une substitution à chaque sous-bloc. Les boîtes de substitution *(S-box)* sont habituellement définies explicitement par des tables.
- Diff est une fonction rapide à calculer, qui effectue la diffusion.



On ajoute après le dernier tour une dernière sous-clef, sans laquelle les opérations de substitution et de diffusion du dernier tour seraient inutiles.

Les opérations de substitution et de diffusion étant inversibles, la fonction de déchiffrement est analogue à la fonction de chiffrement.

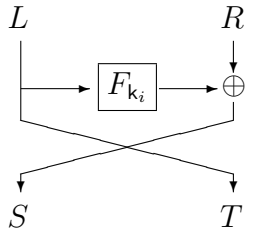
Réseaux de substitution-permutation. On préfère des étapes de substitution et de diffusion qui commutent, ce qui facilite l'implantation dans un même système de l'algorithme de chiffrement et de celui de déchiffrement. Le cas le plus simple est celui des réseaux de substitution-permutation *(SP-networks)* où l'application Diff est une permutation des bits. Mais la diffusion créée par une permutation des bits n'est pas très efficace et il faut beaucoup de tours pour obtenir une sécurité raisonnable.

L'exemple de Rijndael. L'algorithme Rijndael [14] (sélectionné comme AES) est basé sur la *Wide Trail Strategy* formalisée par Daemen [12], où l'étape de diffusion est choisie pour maximiser le nombre de boîtes de substitution mises en relation. Les boîtes de substitution ont une entrée et une sortie de 8 bits, ce qui est un bon compromis permettant une implantation sur carte à puce tout en obtenant une bonne sécurité après une dizaine de tours. C'est une unique boîte de substitution qui est utilisée 16 fois en parallèle.

L'étape de diffusion est en réalité la succession d'une opération appelée `ShiftRow` et d'une opération `MixColumn`. Le bloc de 16 octets est représenté par une matrice 4×4. L'opération `ShiftRow` décale les lignes de la matrice, d'un offset différent pour chaque ligne. L'opération `MixColumn` est la principale nouveauté de Rijndael. Si une colonne est vue comme un polynôme à coefficients dans $GF(2^8)$, alors cette colonne est transformée par multiplication modulo $x^4 + 1$ par un certain polynôme fixé.

2.2.6 Réseaux de type Feistel

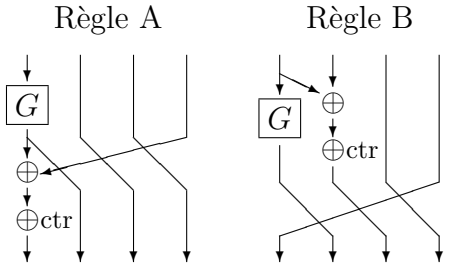
Principe de construction. Chaque tour découpe le message en deux, calcule l'image d'une moitié par une fonction F_{k_i}, ajoute par ou exclusif cette valeur à l'autre moitié, et échange les deux moitiés. Avantages : la fonction F_{k_i} n'a pas besoin d'être bijective, le déchiffrement se fait de la même façon que le chiffrement.



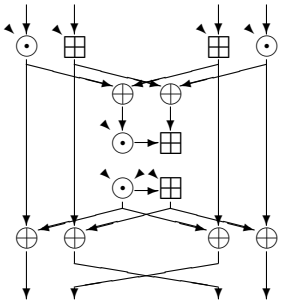
L'exemple de DES. C'est un schéma de Feistel à 16 tours. La fonction F est un réseau à base de boites de substitution. Plus précisément, le demi-bloc (qui fait 32 bits) est étendu en un mot de 48 bits, auquel la sous-clef est ajoutée. Puis il passe à travers 8 boîtes $6 \to 4$ bits, puis une permutation des 32 bits obtenus. Les 8 boîtes de substitution sont distinctes. Leur définition peut être trouvée dans [27], et a suscité beaucoup d'interrogations.

2.2.7 Autres constructions

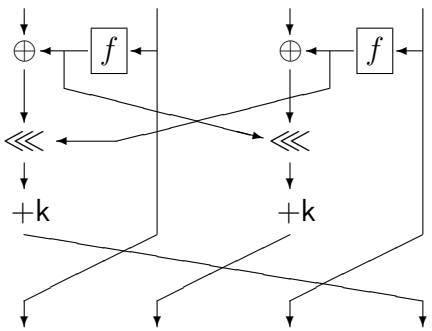
La structure de Skipjack. L'algorithme Skipjack découpe le bloc en 4 mots de 16 bits. Il utilise deux types de tours, appelés A et B. L'exécution de l'algorithme consiste en 8 tours A, puis 8 tours B, puis 8 tours A, puis 8 tours B. Pour chaque tour de Skipjack, une fonction G modifie un quart du bloc. Elle est définie par un Feistel à quatre tours basé sur une unique fonction F, elle-même définie par une table.



La structure d'IDEA. L'algorithme IDEA chiffre des blocs de $64 = 4n$ bits en utilisant trois opérations arithmétiques : $\oplus$ est le ou exclusif de n bits ; $\boxplus$ est l'addition modulo 2^n ; $\odot$ est la multiplication modulo $2^n + 1$, 0 valant 2^n.
Il comporte 8 tours identiques, plus une addition finale d'une sous-clef. Le déchiffrement est similaire.



La structure de RC6. Cet algorithme est un successeur de RC5 et chiffre des blocs de $128 = 4w = 2^{k+2}$ bits en utilisant six opérations arithmétiques : $\oplus$ est le ou exclusif de w bits ; $+$ et $-$ sont l'addition et la soustraction modulo 2^w ; $\times$ est la multiplication modulo 2^w ; $\lll$ et $\ggg$ sont des rotations de mots de w bits. Il comporte 20 tours.
La multiplication est utilisée dans le calcul de $f(x) = (x(2x+1) \bmod 2^w) \lll k$. Les opérations de rotation d'offset variable sont efficaces sur des microprocesseurs Pentium, mais posent problème pour la plupart des autres architectures.



2.3 Étude théorique des schémas de Feistel

2.3.1 Attaques utilisant un distingueur

Définition 2.5 (Distingueur) *Un distingueur pour un cryptosystème donné est un attaquant capable de détecter que c'est ce cryptosystème qui est utilisé et non une fonction aléatoire. Le distingueur a éventuellement accès à un oracle de chiffrement ou de déchiffrement.*

Si le cryptosystème comporte r tours, à l'aide d'un distingueur pour $r-1$ tours, on peut retrouver par recherche exhaustive la sous-clef du dernier tour. Parmi toutes les hypothèses possibles pour cette sous-clef, la bonne est celle pour laquelle le distingueur détecte l'utilisation du cryptosystème.

On suppose que le cryptosystème est un schéma de Feistel sur des blocs de $2n$ bits, pour lequel chaque fonction F est aléatoire. C'est-à-dire que la clef du système à r tours est la description de r fonctions de n bits vers n bits. [5]

On recherche des bornes inférieures et supérieures au nombre de couples clair-chiffré nécessaires pour construire un distingueur d'un schéma de Feistel à r tours. Notons (L, R) le clair et (S, T) le chiffré. La fonction F du k-ième tour est notée F_k.

2.3.2 Attaques génériques

- **1 tour — 1 clair connu et $\mathcal{O}(1)$ opérations.** À l'aide d'un clair connu, on vérifie que la moitié gauche du clair égale la moitié droite du chiffré.
- **2 tours — 2 clairs choisis et $\mathcal{O}(1)$ opérations.** On chiffre (L_1, R_1) et (L_2, R_2) tels que $R_1 = R_2$ et on vérifie que $S_1 \oplus L_1 = S_2 \oplus L_2$.
- **2 tours — $\mathcal{O}(2^{n/2})$ clairs aléatoires et $\mathcal{O}(2^{n/2})$ opérations.** Pour $\mathcal{O}(2^{n/2})$ messages aléatoires, il existe une paire (L_i, R_i) et (L_j, R_j) tels que $R_i = R_j$. On vérifie alors que $S_i \oplus L_i = S_j \oplus L_j$.
- **3 tours — 2 clairs choisis, 1 chiffré choisi et $\mathcal{O}(1)$ opérations.** On chiffre $(L_1, R) \to (S_1, T_1)$ et $(L_2, R) \to (S_2, T_2)$ puis on déchiffre $(S_1 \oplus L_1 \oplus L_2, T_1) \to (L_3, R_3)$. On vérifie alors que $R_1 \oplus R_3 = T_1 \oplus T_2$.
- **3 tours — $\mathcal{O}(2^{n/2})$ clairs choisis et $\mathcal{O}(2^{n/2})$ opérations.** Pour $\mathcal{O}(2^{n/2})$ messages avec L_i constant, il y a deux fois plus de collisions $R_i \oplus S_i = R_j \oplus S_j$ pour un schéma de Feistel à 3 tours avec fonctions F aléatoires que pour une permutation aléatoire. En effet, $R_i \oplus S_i = F_2(F_1(R_i))$, ce qui ajoute aux collisions de F_1 celles de $F_2 \circ F_1$. Autre attaque : si R_i est constant, alors toute collision $T_i = T_j$ est une collision $L_i \oplus S_i = L_j \oplus S_j$ et réciproquement.
- **4 tours — $\mathcal{O}(2^{n/2})$ clairs choisis et $\mathcal{O}(2^{n/2})$ opérations.** Pour $\mathcal{O}(2^{n/2})$ messages avec R_i constant, il y a deux fois plus de collisions $L_i \oplus S_i = L_j \oplus S_j$ pour un schéma de Feistel à 4 tours avec fonctions F aléatoires que pour une permutation aléatoire. En effet, si on note $C = F_1(R_i)$ (valeur constante), on a $L_i \oplus S_i = F_3(F_2(L_i \oplus C)) \oplus C$, ce qui ajoute aux collisions de F_2 celles de $F_3 \circ F_2$.
- **5 tours.** Patarin [37] décrit une attaque nécessitant $\mathcal{O}(2^{3n/2})$ clairs choisis et $\mathcal{O}(2^{3n/2})$ opérations, et une attaque nécessitant $\mathcal{O}(2^{7n/4})$ clairs aléatoires et $\mathcal{O}(2^{7n/4})$ opérations.
- **6 tours et plus.** On ne connaît pas d'attaque nécessitant moins de $\mathcal{O}(2^{2n})$ couples clairs-chiffrés et ayant une complexité meilleure que $\mathcal{O}(2^{2n})$. Remarquons cependant

[5] La description d'une fonction de n bits vers n bits nécessite $n2^n$ bits, la taille de la clef est donc $rn2^n$.

qu'avec $\mathcal{O}(2^n)$ clairs choisis, une recherche exhaustive sur les fonctions F_k donne une attaque en $2^{\mathcal{O}(n2^n)}$ opérations

2.3.3 Preuves de sécurité

Si la puissance de calcul de l'attaquant est infinie, il existe donc une attaque demandant $\mathcal{O}(2^n)$ clairs choisis, quelque soit le nombre de tours. Le problème reste ouvert de savoir s'il existe un nombre minimal de tours à partir duquel toute attaque nécessite $\mathcal{O}(2^n)$ messages.

Luby et Rackoff ont montré en 1988 [25] qu'une attaque à clairs choisis d'un schéma de Feistel d'au moins 3 tours demande au moins $\mathcal{O}(2^{n/2})$ clairs, et qu'une attaque à clairs et chiffrés choisis d'un schéma de Feistel d'au moins 4 tours demande au moins $\mathcal{O}(2^{n/2})$ messages.

Ce résultat a inspiré de nombreux développements, en particulier les résultats de Patarin [35, 36] sur un plus grand nombre de tours. Toujours avec une puissance de calcul illimitée, l'attaque d'un schéma de Feistel à 5 tours demande au moins $\mathcal{O}(2^{2n/3})$ messages, et attaque d'un schéma de Feistel à 6 tours demande au moins $\mathcal{O}(2^{3n/4})$ messages.

2.4 Cryptanalyse différentielle

2.4.1 Description de l'attaque

Biham et Shamir ont révélé au congrès Crypto'90 et détaillé et amélioré ultérieurement [3, 5, 6] une technique de cryptanalyse du DES qu'ils ont appelé *cryptanalyse différentielle*. Ils ont au passage montré que DES est particulièrement bien protégé contre ce type d'attaque, ce qui laisse penser que les concepteurs du DES l'avaient prévue.

C'est une attaque à clair choisi, adaptée aux systèmes de chiffrement itératifs, où on fabrique un distingueur.

Différentielle et attaque par distingueur.

Une différentielle est la donnée de deux valeurs δ et δ^* dans l'espace des blocs. La différentielle est vérifiée pour la clef k et la paire (x, x') si $x \oplus x' = \delta$ et $\mathsf{E_k}(x) \oplus \mathsf{E_k}(x') = \delta^*$. La **probabilité de la différentielle** est la probabilité p sur l'ensemble des clefs et sur l'ensemble des paires telles que $x \oplus x' = \delta$ que la différentielle soit vérifiée. Ceci est noté $\delta \rightarrow_p \delta^*$.

On note aussi p_k la probabilité lorsque la clef est fixée et $p_{x,x'}$ lorsque la paire de textes clairs est fixée.

Le principe de l'attaque par distingueur est le suivant : l'attaquant chiffre des paires (x, x') de différence δ et observe le résultat (y, y'). Appelons p^* la probabilité qu'une paire aléatoire vérifie $y \oplus y' = \delta^*$. L'attaquant sait que la probabilité que $y \oplus y' = \delta^*$ vaut p_k si le système est bien $\mathsf{E_k}$, et que cette probabilité vaut $p^* = 2^{-b}$ pour une permutation aléatoire. Si $p_\mathsf{k} \neq p^*$, alors le chiffrement d'un nombre N suffisamment élevé de paires aléatoires indépendantes et de différence δ permet de distinguer avec bonne probabilité $\mathsf{E_k}$ d'une permutation aléatoire.

Supposons que $p_\mathsf{k} > p^*$ (l'inégalité inverse se traite de façon similaire) et introduisons un seuil α tel que : si on observe au moins α paires telles que $y \oplus y' = \delta^*$, alors on affirme que la boite noire implante $\mathsf{E_k}$, sinon on affirme que c'est une permutation aléatoire. Une

valeur précise de la probabilité d'échec de ce distingueur peut être obtenue avec les bornes de Chernoff. [6]

Si la probabilité p_{k} est indépendante de k alors le distingueur permet de distinguer avec bonne probabilité le système de chiffrement d'une permutation aléatoire. Si la probabilité p_{k} n'est pas indépendante de k, la même technique permet de détecter des clefs faibles.

Calculs de probabilités.

Il est habituellement impossible, même si la clef k est connue, de calculer la probabilité d'une différentielle par énumération, car il faudrait chiffrer les 2^b paires telles que $x \oplus x' = \delta$. Néanmoins, la régularité de la structure de la fonction E_{k} permet souvent d'évaluer la probabilité d'une différentielle.

Juxtaposition. Si la fonction E_{k} est la juxtaposition de n fonctions (des boîtes de substitution), alors la probabilité de la différentielle est le produit des probabilités correspondantes. Plus précisément, si $\mathsf{E}_{\mathsf{k}}(x_1\|...\|x_n) = \mathsf{E}^1_{\mathsf{k}_1}(x_1)\|...\|\mathsf{E}^n_{\mathsf{k}_n}(x_n)$, alors $\Pr[(\delta_1\|...\|\delta_n) \to (\delta'_1\|...\|\delta'_n)] = \prod_i \Pr[\delta_i \to \delta'_i]$.

Ceci permet de calculer la probabilité d'une différentielle pour un tour d'un réseau de substitution-permutation.

Composition. La plupart des algorithmes de chiffrement étant itératifs, il est important d'avoir une technique pour calculer la probabilité d'une différentielle sur plusieurs tours. C'est cela qui donne un grand avantage à la cryptanalyse différentielle sur des techniques plus anciennes.

Étant données r fonctions paramétrées par des clefs $\mathsf{E}^i_{\mathsf{k}^i} : \{0,1\}^{n_{i-1}} \to \{0,1\}^{n_i}$, on définit $\mathsf{k} = \mathsf{k}^1\mathsf{k}^2...\mathsf{k}^r$ la concaténation des clefs et $\mathsf{E}_{\mathsf{k}} = \mathsf{E}^r_{\mathsf{k}^r} \circ ... \circ \mathsf{E}^2_{\mathsf{k}^2} \circ \mathsf{E}^1_{\mathsf{k}^1}$ la composition de ces fonctions. Les sous-clefs sont donc indépendantes.

On appelle **caractéristique différentielle** à r tours et on note $\delta^0 \to \delta^1 \to ... \to \delta^r$ l'événement vérifié pour la clef k et la paire (x, x') si $x \oplus x' = \delta^0$ et pour tout i on a $(\mathsf{E}^i_{\mathsf{k}^i} \circ ... \circ \mathsf{E}^1_{\mathsf{k}^1})(x) \oplus (\mathsf{E}^i_{\mathsf{k}^i} \circ ... \circ \mathsf{E}^1_{\mathsf{k}^1})(x') = \delta^i$.

Si pour tout $i = 1...r-1$ on partitionne $\{0,1\}^{n_i} = \cup_{j_i}\{\delta^i_{j_i}\}$, la probabilité de la différentielle $\delta^0 \to \delta^r$ est évidemment égale à la somme des probabilités de toutes les caractéristiques différentielles $\delta^0 \to ... \to \delta^i_{j_i} \to ... \to \delta^r$.

Hypothèse 1 (Système de chiffrement markovien) *Une fonction E est dite markovienne pour la différentielle $\delta \to \delta^*$ si la probabilité $p_{x,x'}$ pour une clef uniformément aléatoire et une certaine paire telle que $x \oplus x' = \delta$ que la différentielle soit vérifiée est indépendante du choix de (x, x').*

---

[6] **Bornes de Chernoff.** Si un événement qui apparaît avec probabilité p est répété avec N tirages indépendants, on appelle X la variable aléatoire qui compte le nombre d'occurences de l'événement. Les bornes de Chernoff mesurent la probabilité que X s'écarte de son espérance Np.
$\Pr[X < (1-\delta)Np] < \left(e^{-\frac{\delta^2}{2}}\right)^{Np}$ et $\Pr[X > (1+\delta)Np] < \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^{Np}$.
La valeur optimale pour α peut être calculée en fonction de $\beta = p_{\mathsf{k}}/p^*$: on prend $\alpha = dNp^*$, avec d solution de $d - 1 - d\ln d + \beta(1-d/\beta)^2/2 = 0$. La probabilité que le distingueur se trompe est alors inférieure à 2ε, avec $\varepsilon = \varepsilon_0^{Np^*}$ où $\varepsilon_0 = e^{d-1}/d^d$.
Par exemple, si $\beta = 2$, alors $d \simeq 1.43$ et $\varepsilon_0 \simeq 0.92$; si $\beta = 2^8$ alors $d \simeq 36$ et $\varepsilon_0 \simeq 2^{-136}$.

Cette hypothèse a d'abord été formalisée par Lai, Massey et Murphy [24], qui remarquent que si toutes les E^i sont des fonctions markoviennes pour $\delta^{i-1} \rightarrow \delta^i$, alors la probabilité de la caractéristique différentielle $\delta^0 \rightarrow \delta^1 \rightarrow ... \rightarrow \delta^r$ est égale au produit des probabilités des différentielles $\delta^0 \rightarrow \delta^1$, $\delta^1 \rightarrow \delta^2$, ... et $\delta^{r-1} \rightarrow \delta^r$. Cela leur permet d'énoncer des bornes inférieures pour la probabilité d'une différentielle pour un système de chiffrement markovien.

Par exemple, les calculs de probabilités de caractéristiques différentielles fait par Biham et Shamir sont valides, puisqu'un tour de DES est une fonction markovienne pour toutes les différentielles $\delta \rightarrow \delta^*$ et qu'ils étudient le cas du DES avec des sous-clefs indépendantes.

Dans le cas où les fonctions E^i sont identiques (système itératif à sous-clefs indépendantes), on peut considérer la matrice de transition M, de dimension 2^b, qui donne les probabilités des différentielles à 1 tour. Les probabilités des différentielles à r tours sont données par M^r, pour un système de chiffrement Markovien. Néanmoins, la dimension de la matrice M empêche un calcul exact...

Hypothèse 2 (Caractéristique prédominante) *Si la probabilité de la caractéristique $\delta^0 \rightarrow \delta^1 \rightarrow ... \rightarrow \delta^r$ est élevée, alors elle est presque égale à la probabilité de la différentielle $\delta^0 \rightarrow \delta^r$.*

L'hypothèse de la caractéristique prédominante n'est donc pas vérifiée si cette caractéristique n'est pas la seule allant de δ^0 à δ^r ayant une probabilité élevée, ce qui est souvent le cas.

Comme le succès d'une attaque dépend de la probabilité de la meilleure différentielle, mais qu'il est plus facile de calculer la probabilité d'une caractéristique, certains auteurs de cryptosystèmes basent à tort la sécurité de leur primitive sur la probabilité de la plus probable caractéristique. C'est par exemple l'une des erreurs qu'a commises le concepteur de l'algorithme Q [2].

2.4.2 Mise en œuvre

Trouver la sous-clef des derniers tours.

C'est le principe de l'attaque de Biham et Shamir, qui leur a permis de casser diverses variantes du DES. On commence par décomposer la fonction $\mathsf{E}_\mathsf{k} = \mathsf{E}^2_{\mathsf{k}_2} \circ \mathsf{E}^1_{\mathsf{k}_1}$. L'attaque utilise une différentielle $\delta \rightarrow_p \delta^*$ pour E^1. Pour chaque paire telle que $x \oplus x' = \delta$, on appelle $y = \mathsf{E}^1_{\mathsf{k}_1}(x)$ et $y' = \mathsf{E}^1_{\mathsf{k}_1}(x')$. Si on suppose que la paire suit la différentielle (on dit dans ce cas qu'il s'agit d'une **bonne paire**, et celles qui ne suivent pas la différentielle sont appelées **mauvaises paires**), alors $y \oplus y' = \delta^*$ et de plus les valeurs $z = \mathsf{E}^2_{\mathsf{k}_2}(y) = \mathsf{E}_\mathsf{k}(x)$ et $z' = \mathsf{E}^2_{\mathsf{k}_2}(y') = \mathsf{E}_\mathsf{k}(x')$ sont connues.

Les valeurs acceptables pour k_2 sont les valeurs k telles que $(\mathsf{E}^2_k)^{-1}(z) \oplus (\mathsf{E}^2_k)^{-1}(z') = \delta^*$. Il se peut qu'aucune valeur k ne soit acceptable, ce qui permet de détecter que la paire (x, x') ne suit pas la différentielle, on rejette donc cette mauvaise paire, ce qui est appelé **filtrage**. Selon la terminologie de Lai, Massey et Murphy [24], la fonction E^2 est dite **cryptographiquement faible** si le calcul des valeurs acceptables pour k_2 est faisable de façon efficace.

Si toutes les paires sont de bonnes paires, alors la clef k_2 est nécessairement l'une des valeurs acceptables pour toutes les paires. S'il reste des mauvaises paires qui n'ont pas été éliminées par le filtrage, la technique de comptage décrite plus en détails dans la section

suivante sert à déduire la bonne valeur de k_2 à partir des valeurs connues (z, z') et de la probabilité p que $y \oplus y' = \delta^*$.

Ensuite, il suffit d'appliquer la même attaque pour trouver la sous-clef des derniers tours de $\mathsf{E}^1_{\mathsf{k}_1}$, et d'avoir progressivement la valeur de toutes les sous-clefs, ce qui ne permet pas nécessairement de retrouver k, mais permet de calculer E_k et donc D_k.

Hypothèse 3 (Équivalence stochastique) *Comme l'attaque doit fonctionner pour une clef k_1 inconnue, on suppose que pour la plupart des clefs on a $p_{\mathsf{k}_1} \simeq p$.*

Cette hypothèse a été formalisée et nommée par Lai, Massey et Murphy [24] et étudiée plus longuement par Lai dans sa thèse [22]. Elle est vérifiée en pratique lorsque le système a un nombre de tours suffisamment élevé.

Biham et Shamir s'affranchissent de cette hypothèse et améliorent leur attaque du DES en utilisant simultanément deux différentielles de même probabilité p telles que pour toute clef k_1 l'une au moins ait $p_{\mathsf{k}_1} \geq p$. Leur attaque utilise donc des quartets de textes choisis, et non plus des paires.

Comptage.

On utilise un tableau de compteurs pour toutes les valeurs possibles de k_2, et on incrémente les compteurs pour toutes les valeurs acceptables suggérées par chaque paire. Pour chaque bonne paire (donc avec probabilité p), la bonne valeur de k_2 fait partie des suggestions émises. Lorsque les autres suggestions émises sont aléatoires et uniformément réparties, et alors la bonne valeur de k_2 est la plus fréquente.

Hypothèse 4 (Comptage) *Après analyse d'un nombre de paires suffisamment élevé, la bonne valeur de la sous-clef correspond à l'un des compteurs ayant la valeur la plus élevée.*

L'hypothèse de comptage n'est habituellement pas vérifiée par les articles publiés, bien qu'on puisse construire de nombreux contre-exemples. L'hypothèse de comptage est vérifiée en pratique avec probabilité élevée si les fonctions E^1 et E^2 ressemblent à des permutations aléatoires.

Les détails de la technique de comptage sont explicités par Biham et Shamir dans la description complète de leur cryptanalyse du DES [3]. Ils définissent en particulier un rapport signal/bruit, qui est le rapport entre le nombre de bonnes paires et la valeur moyenne d'un compteur. Ce rapport, qui est indépendant du nombre de paires étudiées, donne une estimation du nombre de bonnes paires nécessaires.

Biham et Shamir proposent aussi diverses améliorations de la technique de comptage, pour économiser de la mémoire, en ne recensant dans les compteurs que certains bits de la clef ou en décrivant les valeurs de clef suggérées sous la forme d'un graphe dont on cherche la plus grande clique [3]. Ils proposent aussi une technique où les mauvaises clefs sont éliminées au fur et à mesure, lorsqu'on est capable de proposer une valeur pour toute la clef k [5].

Trouver la sous-clef des premiers tours.

Bien sûr, on peut trouver la sous-clef des premiers tours à l'aide d'une attaque à chiffré choisi de la même façon qu'on trouve la sous-clef des derniers tours avec une attaque à clair choisi. Mais il existe aussi une attaque à clair choisi, décrite par exemple par Knudsen, Robshaw et Wagner dans [20].

On commence par décomposer la fonction $\mathsf{E_k} = \mathsf{E}^2_{\mathsf{k}_2} \circ \mathsf{E}^1_{\mathsf{k}_1}$. On utilise trois différentielles $\delta \rightarrow_q \delta^\times$ pour $\mathsf{E}^1_{\mathsf{k}_1}$ (les quelques premiers tours), $\delta^\times \rightarrow_p \delta^*$ pour $\mathsf{E}^2_{\mathsf{k}_2}$ (le reste du système) et $\delta \rightarrow_{p^*} \delta^*$ pour $\mathsf{E_k}$ (tout le système).

L'attaque crée des paires aléatoires vérifiant δ, et ne garde que celles qui après chiffrement vérifient δ^* (filtrage). On utilise un tableau de compteurs pour toutes les valeurs possibles de k_1, et on incrémente les compteurs pour toutes les valeurs de la clef telles que après chiffrement par $\mathsf{E}^1_{\mathsf{k}_1}$ la paire vérifie $\delta^\times$. L'attaque est réussie si le compteur le plus élevé correspond à la bonne sous-clef.

Sans filtrage, la valeur du compteur associé à chaque valeur pour k_1 est proportionnelle à la valeur de la probabilité q_{k_1} que la différentielle $\delta \rightarrow \delta^\times$ soit vérifiée pour k_1. Si la probabilité conditionnelle $q^* = \Pr[\delta \rightarrow \delta^\times \rightarrow \delta^* / \delta \rightarrow \delta^*]$ est supérieure à la probabilité $q = \Pr[\delta \rightarrow \delta^\times]$, alors le filtrage augmente la probabilité que la paire vérifie effectivement $\delta^\times$ après les premiers tours, et la valeur du compteur associé à la bonne valeur de k_1 est augmentée par le filtrage.

La non vérification de cette hypothèse est la principale erreur de [20] relevée dans [15]. En effet, si on suppose que le système est markovien, alors la probabilité q^* (pour des clefs aléatoires et indépendante) est égale au quotient de $\Pr[\delta \rightarrow \delta^\times \rightarrow \delta^*]$ par $\Pr[\delta \rightarrow \delta^*]$, c'est-à-dire à pq/p^*, qui doit être supérieure à q. Une propriété nécessaire pour que l'attaque puisse réussir est donc $p > p^*$.

2.4.3 Résistance prouvée

L'article de Lai, Massey et Murphy [24], en introduisant la notion de système de chiffrement markovien, permet de donner des bornes inférieures pour la probabilité d'une différentielle, lorsqu'on sait calculer toutes les probabilités des caractéristiques différentielles non négligeables. Le problème se pose de savoir quelles caractéristiques différentielles négliger.

Pour une caractéristique donnée, si on appelle *boîtes actives* les boîtes de substitution pour lesquelles la différence est non nulle, la probabilité de la caractéristique est en gros le produit des probabilités de ces différences. On néglige donc les caractéristiques différentielles pour lesquelles le nombre de boîtes actives est élevé.

Une étude a été développée par Nyberg et Knudsen [34] pour le cas des schémas de Feistel, mais il apparaît qu'un système se basant uniquement sur la résistance à la cryptanalyse différentielle est très susceptible à d'autres attaques...

La théorie de la décorrélation a été inventée par Vaudenay pour établir des preuves de résistance à la cryptanalyse pour les systèmes de chiffrement itératifs.

2.5 Variantes de la cryptanalyse différentielle

Remarquons que la définition d'une différentielle n'utilise pas le fait que $\mathsf{E_k}$ soit bijective. Le cryptanalyse différentielle peut donc par exemple s'appliquer aux fonctions de hachage.

En outre, le test d'égalité $x \oplus x' = \delta$ peut être remplacé par n'importe quel test $\mathsf{T}(x, x')$, et similairement $\mathsf{T}^*(\mathsf{E_k}(x), \mathsf{E_k}(x'))$. On a un distingueur dès que la probabilité $\mathsf{T} \rightarrow \mathsf{T}^*$ est non triviale. Mais cette probabilité n'est facilement calculable que si le système est markovien. En pratique, le test T est toujours défini par $x \boxplus x'^{-1} = \delta$ pour une loi de groupe $\boxplus$.

2.5.1 Différentielles tronquées

Introduites par Knudsen [18], les différentielles tronquées rajoutent une souplesse : seuls certains bits de la différence sont imposés. Cela correspond à considérer deux ensembles Δ et Δ^* et la probabilité que $\mathsf{E}_\mathsf{k}(x) \oplus \mathsf{E}_\mathsf{k}(x') \in \Delta^*$ lorsque $x \oplus x' \in \Delta$.

Attention : un système de chiffrement peut être markovien pour la cryptanalyse différentielle classique, mais ne pas l'être pour la cryptanalyse différentielle tronquée. Cela complique les calculs de probabilité, cf. [15].

De plus, contrairement au cas des singletons, la probabilité de $\Delta \to \Delta^*$ pour E n'est pas égale à la probabilité de $\Delta^* \to \Delta$ pour D.

2.5.2 Différentielles impossibles

Ce concept a été mis en œuvre par Biham, Biryukov et Shamir [1] pour une cryptanalyse de Skipjack. Le principe de l'attaque est de trouver une différentielle $\delta \to \delta^*$ ayant probabilité 0. Ceci permet de construire un distingueur particulièrement efficace.

La meilleure façon de trouver une différentielle de probabilité 0 pour un cryptosystème découpé en $\mathsf{E} = \mathsf{E}_2 \circ \mathsf{E}_1$ est de trouver une différentielle $\delta \to_1 \delta'$ pour E_1 et une différentielle $\delta^\times \to_1 \delta^*$ pour E_2, toutes deux de probabilité 1, avec $\delta' \neq \delta^\times$.

L'attaque de Biham, Biryukov et Shamir sur 31 tours de Skipjack utilise une différentielle impossible sur 24 tours pour deviner les sous-clefs des cinq autres. Les détails pratiques de cette attaque sont comparables aux détails pratiques de l'attaque du DES par caractéristiques différentielles de probabilité élevée.

2.5.3 Différentielles d'ordre supérieur

Lai a remarqué que si on fixe une loi de groupe + sur l'espace des blocs, alors la dérivée d'une fonction f est définie par la valeur de $D_\delta(f)(x) = f(x+\delta) - f(x)$ et la dérivée d'ordre n est définie par $D^{(n)}_{\delta_1,\ldots,\delta_n}(f)(x) = D_{\delta_n}(D^{(n-1)}_{\delta_1,\ldots,\delta_{n-1}}(f))(x) = \sum_{I \subset \{1,\ldots,n\}} (-1)^{n-\#I} f(x + \sum_{i \in I} \delta_i)$. Les différentielles de Biham et Shamir s'expriment alors sous la forme $D_\delta(\mathsf{E}_\mathsf{k})(x) = \delta^*$ pour la loi de groupe $\oplus$, que Lai généralise avec les **différentielles d'ordre supérieur** $(\delta_1, ..., \delta_n) \to \delta^*$ utilisant donc des 2^n-uplets de clairs choisis [23, 18].

2.5.4 Boomerang

Introduite par Wagner [43], l'attaque boomerang est une mise en œuvre pratique d'une cryptanalyse à base de quadruplets, qu'on peut voir comme un cas particulier d'attaque différentielle d'ordre 2. Elle est donc très efficace contre les algorithmes ne se protégeant que contre les attaques à base de paires. C'est une attaque à clairs et à chiffrés choisis.

Le cryptosystème est découpé en $\mathsf{E} = \mathsf{E}_2 \circ \mathsf{E}_1$. L'attaque utilise une différentielle $\delta \to_p \delta^*$ pour E_1 et la différentielle $\gamma \to_q \gamma^*$ pour E_2^{-1}. Deux textes tels que $x \oplus x' = \delta$ sont chiffrés par E pour donner c et c'. Deux textes d et d' sont choisis tels que $c \oplus d = \gamma$ et $c' \oplus d' = \gamma$, et sont déchiffrés par E^{-1} pour donner y et y'. Notons $\bar{x}$, $\bar{x}'$, $\bar{y}$ et $\bar{y}'$ le chiffrement par E_1 de x, x', y, et y'. On s'intéresse aux occurrences de l'événement $\mathcal{E}$ défini par les quatre équations : $\bar{x} \oplus \bar{x}' = \delta^*$, $\bar{y} \oplus \bar{y}' = \delta^*$, $\bar{x} \oplus \bar{y} = \gamma^*$ et $\bar{x}' \oplus \bar{y}' = \gamma^*$. Trois des équations impliquent la quatrième. La probabilité que cet événement survienne est donc pq^2, et donc la probabilité d'avoir simultanément $\mathcal{E}$ er $y \oplus y' = \delta$ est p^2q^2. Ceci peut suffire pour avoir un distingueur du cryptosystème.

Remarquons qu'on peut généraliser l'attaque boomerang aux différentielles tronquées. Il faut alors (comme le remarque Wagner [43]) distinguer la probabilité $\Delta \to \Delta^*$ utilisée à l'aller et la probabilité $\Delta^* \to \Delta$ utilisée au retour. De plus, trois des quatre équations au milieu du système n'impliquent pas nécessairement la quatrième.

2.6 Autres techniques de cryptanalyse

2.6.1 Cryptanalyse linéaire

Introduite par Matsui [26], la cryptanalyse linéaire est similaire à la cryptanalyse différentielle en ce qu'elle tire son efficacité de la recherche de caractéristiques (linéaires) pour des cryptosystèmes itératifs.

Une relation linéaire pour $c = \mathsf{E}_\mathsf{k}(m)$ est définie par trois valeurs δ, δ^* et κ et la relation $m \cdot \delta \oplus c \cdot \delta^* = \mathsf{k} \cdot \kappa$, où $\cdot$ est le produit scalaire. Autrement dit, δ, δ^* et κ sélectionnent des bits des messages et de la clef pour en faire un ou exclusif. Une telle relation est utilisable pour la cryptanalyse si sa probabilité (pour une clef donnée et un message aléatoire) est différente de 1/2. On a ainsi une attaque à clair connu.

Les relations linéaires se combinent par juxtaposition et par composition, ce qui permet donc de trouver des caractéristiques linéaires pour un système complet en étudiant ses composantes.

2.6.2 Attaque par interpolation

Pour résister aux cryptanalyses différentielle et linéaire, il est important que les boîtes de substitution n'aient pas de bonne approximation linéaire. Nyberg [33, 34] a proposé diverses constructions de telles boîtes, à partir de polynômes dans $GF(2^n)$. Jakobsen et Knudsen [16] remarquent que le petit degré de ces polynômes permet de construire une bonne approximation polynomiale du cryptosystème, et donc une attaque.

Les auteurs de Rijndael ont donc modifié les boîtes de substitution de Nyberg pour leur donner un haut degré, tout en leur laissant de bonnes propriétés vis à vis des cryptanalyses différentielle et linéaire.

2.6.3 Attaque par résolutions de système d'équations algébriques

D'abord introduite pour la cryptanalyse de systèmes à clef publique [39], cette technique peut éventuellement s'appliquer à la cryptanalyse de systèmes de chiffrement par bloc [11]. Son principe est que les relations entre clair, chiffré et clef peuvent s'exprimer par un système d'équations sur les bits, c'est-à-dire de polynômes à plusieurs variables dans $GF(2)$. Comme clairs et chiffrés sont connus, les inconnues de ce système sont les bits de clef et la résolution de ce système permet de retrouver la clef. En pratique, les algorithmes de résolution de tels systèmes demandent un temps de calcul élevé et une immense mémoire, mais certains algorithmes de chiffrement par bloc sont peut-être vulnérables.

2.6.4 Attaque par décalage

Introduite par Biryukov et Wagner [8], cette technique est dédiée aux cryptosystèmes ayant un grand nombre de tours, car contrairement aux autres techniques de cryptanalyse

son efficacité ne dépend pas du nombre de tours. Cette attaque s'applique aux cryptosystèmes dont un tour est facilement susceptible à une cryptanalyse avec deux clairs connus, et dont la diversification de clef est extrêmement simple, par exemple avec des sous-clefs périodiques.

Si tous les tours sont identiques et avec la même sous-clef ($\mathsf{E_k} = \mathsf{F_k} \circ ... \circ \mathsf{F_k}$), alors le chiffrement $c = \mathsf{E_k}(m)$ est équivalent à $c' = \mathsf{E_k}(m')$ avec $m' = \mathsf{F_k}(m)$ et $c' = \mathsf{F_k}(c)$. Les paires $(m'c)$ et (m', c') sont décalées d'un tour. Il suffit de $2^{b/2}$ couples clairs-chiffrés pour avoir des paires décalées, qui peuvent être isolées en détectant s'il est possible qu'il existe une clef telle que $m_j = \mathsf{F_k}(m_i)$ et $c_j = \mathsf{F_k}(c_i)$. Par exemple, pour un schéma de Feistel, ceci est détecté en triant les messages par la valeur de leur moitié non modifiée par F.

Biryukov et Wagner ont aussi décrit des améliorations de cette technique pour une plus grande variété de cryptosystèmes [9].

2.6.5 Attaque par saturation

Aussi appelée *attaque structurelle* [7] ou *cryptanalyse intégrale* [21] ou bien encore *Square attack*, car cette technique est apparue avec l'algorithme Square [13], précurseur de Rijndael.

Il s'agit d'étudier le comportement du cryptosystème lorsqu'on chiffre une grande famille de messages, choisie pour que certaines composantes du système prennent toutes les valeurs possibles. L'exemple de la cryptanalyse de SASAS permet d'illustrer cette technique.

Attaque de SASAS. Le cryptosystème SASAS est un système théorique qui chiffre un message de $b = km$ bits en cinq étapes. La première opération (appelée S) est une substitution parallèle du bloc découpé en k sous-blocs de m bits. La seconde opération (appelée A) est une application affine sur le bloc de b bits, vu comme un vecteur sur $GF(2)$. Et ainsi de suite jusqu'à obtenir cinq étapes : SASAS. On considère que la description des opération S et A fait partie de la clef, qui est donc longue de $3k \log_2(2^m!) - 3.57 + 2n^2 + 2n$ bits, ce qui fait environ 110000 bits pour $m = 8$ et $n = 128$.

La cryptanalyse de SASAS est une attaque à clair choisi, où on chiffre un ensemble de 2^m messages, ayant la propriété $\mathbf{C}^{i-1}\mathbf{P}\mathbf{C}^{k-i}$. Cela signifie que lorsqu'on découpe ces messages en k sous-blocs de m bits, on obtient k multiensembles[7] ayant la propriété $\mathbf{C}$, le i-ième ayant la propriété $\mathbf{P}$. Les cinq propriétés des multiensembles de mots de m bits qui sont utiles pour la cryptanalyse de SASAS sont les suivantes.
Propriété $\mathbf{C}$ (constant) s'il contient un nombre arbitraire de fois une unique valeur.
Propriété $\mathbf{P}$ (permutation) s'il contient une fois chacune des 2^m valeurs.
Propriété $\mathbf{E}$ (even/pair) si chaque élément y apparaît un nombre pair de fois.
Propriété $\mathbf{B}$ (balanced/équilibré) si le XOR de toutes ses valeurs (avec multiplicité) est 0.
Propriété $\mathbf{D}$ (dual) s'il a l'une des propriétés $\mathbf{P}$ ou $\mathbf{E}$.

On peut montrer que l'opération S conserve les propriétés $\mathbf{C}$, $\mathbf{P}$ et $\mathbf{E}$, car pour chaque sous-bloc c'est une permutation des 2^m valeurs possibles. De plus, l'opération P conserve $\mathbf{B}^k$ si le nombre d'éléments du multiensemble est pair, donc transforme $\mathbf{D}^k$ en $\mathbf{B}^k$. Enfin, l'opération P transforme $\mathbf{C}^{i-1}\mathbf{P}\mathbf{C}^{k-i}$ en $\mathbf{D}^k$.

On en déduit donc un distingueur de SASA, qui chiffre un ensemble de messages ayant la propriété $\mathbf{C}^{i-1}\mathbf{P}\mathbf{C}^{k-i}$ en un ensemble de chiffrés ayant la propriété $\mathbf{B}^k$. Ce distingueur permet de retrouver le dernier tour de SASAS. L'attaque complète a été implantée par Biryukov et Shamir [7].

[7] Un multiensemble est un ensemble dont les éléments peuvent être répétés.

Bibliographie

[1] E. Biham, A. Biryukov et A. Shamir. – Cryptanalysis of skipjack reduced to 31 rounds using impossible differentials. – *In : Advances in Cryptology, proceedings of Eurocrypt'99*, Prague, République Tchèque, 2–6 mai 1999, éd. par J. Stern. – LNCS, vol. 1592, pp. 12–23. Springer-Verlag, 1999. Cf. `http://www.cs.technion.ac.il/users/wwwb/cgi-bin/tr-get.cgi/1998/CS/CS0947.revised.ps.gz`

[2] E. Biham, V. Furman, M. Misztal et V. Rijmen. – Differential cryptanalysis of Q. – *In : Fast Software Encryption : 8th International Workshop*, Yokohama, Japon, 2–4 avr. 2001, éd. par M. Matsui. – Springer-Verlag, 2001.

[3] E. Biham et A. Shamir. – Differential cryptanalysis of DES-like cryptosystems. *Journal of Cryptology*, vol. 4, n° 1, 1991, pp. 3–72. Une version abrégée est parue dans [4]. Cf. `http://www.cs.technion.ac.il/~biham/Reports/Weizmann/cs90-16.ps.gz`

[4] E. Biham et A. Shamir. – Differential cryptanalysis of DES-like cryptosystems. – *In : Advances in Cryptology, proceedings of Crypto'90*, Santa Barbara, CA, 11–15 août 1990, éd. par A. Menezes et S. A. Vanstone. – LNCS, vol. 537, pp. 2–21. Springer-Verlag, 1991.

[5] E. Biham et A. Shamir. – Differential cryptanalysis of the full 16-round DES. – *In : Advances in Cryptology, proceedings of Crypto'92*, Santa Barbara, CA, 16–20 août 1992, éd. par E. F. Brickell. – LNCS, vol. 740, pp. 487–496. Springer-Verlag, 1993. Cf. `http://www.cs.technion.ac.il/users/wwwb/cgi-bin/tr-get.cgi/1991/CS/CS0708.ps.gz`

[6] E. Biham et A. Shamir. – *Differential Cryptanalysis of the Data Encryption Standard*. – Springer-Verlag, 1993.

[7] A. Biryukov et A. Shamir. – Structural cryptanalysis of SASAS. – *In : Advances in Cryptology, proceedings of Eurocrypt'01*, Innsbruck, Autriche, 7–10 mai 2001, éd. par B. Pfitzmann. – LNCS, vol. 2045, pp. 394–405. Springer-Verlag, 2001. Cf. `http://www.wisdom.weizmann.ac.il/~albi/eu01sasas.ps.gz`

[8] A. Biryukov et D. Wagner. – Slide attacks. – *In : Advances in Cryptology, proceedings of FSE'99*, Rome, Italie, 24–26 mars 1999, éd. par L. Knudsen – LNCS, vol. 1636. Springer-Verlag, 1999. Cf. `http://www.cs.berkeley.edu/~daw/papers/slide-fse99.ps`.

[9] A. Biryukov et D. Wagner. – Advanced slide attacks. – *In : Advances in Cryptology, proceedings of Eurocrypt'00*, Bruges, Belgique, 14–18 mai 2000, éd. par B. Preneel. – LNCS, vol. 1807. Springer-Verlag, 2000. Cf. `http://www.cs.berkeley.edu/~daw/papers/advslide-ec00.ps`.

[10] J. Black, S. Halevi, H. Krawczyk, T. Krovetz et P. Rogaway. – UMAC : Fast and Secure Message Authentication. – *In : Advances in Cryptology, proceedings of Crypto'99*,

Santa Barbara, Californie, 15–19 août 1999, éd. par M. Wiener. – LNCS, vol. 1666, pp. 216–233. Springer-Verlag, 1999. Cf. http://www.cs.ucdavis.edu/~rogaway/umac/

[11] N. Courtois et J. Pieprzyk. – Cryptanalysis of block ciphers with overdefined systems of equations. – *In : Advances in Cryptology, proceedings of Asiacrypt'02*, éd. par Y. Zheng. – LNCS, vol. 2501, pp. 267–287. Springer-Verlag, 2002.

[12] J. Daemen. – *Cipher and hash function design strategies based on linear and differential cryptanalysis.* – Thèse, K.U.Leuven, mars 1995.

[13] J. Daemen, L. Knudsen et V. Rijmen. – The block cipher Square. – *In : Fast Software Encryption : Fourth International Workshop*, Haifa, Israel, 20–22 jan. 1997, éd. par E. Biham. – LNCS, vol. 1267, pp. 149–165. Springer-Verlag, 1997. Cf. http://www.esat.kuleuven.ac.be/~rijmen/square/

[14] J. Daemen et V. Rijmen. – AES proposal : Rijndael. – *AES algorithm submission*, 1999. Cf. http://www.esat.kuleuven.ac.be/~rijmen/rijndael/

[15] L. Granboulan. – Flaws in differential cryptanalysis of Skipjack. – *In : Fast Software Encryption : 8th International Workshop*, Yokohama, Japon, 2–4 avr. 2001, éd. par M. Matsui. – Springer-Verlag, 2001. Cf. http://eprint.iacr.org/2001/038/

[16] T. Jakobsen et L. Knudsen. – The Interpolation Attack on Block Ciphers. – *In : Fast Software Encryption : Fourth International Workshop*, Haifa, Israel, 20–22 jan. 1997, éd. par E. Biham. – LNCS, vol. 1267. Springer-Verlag, 1997. Cf. http://www.mat.dtu.dk/persons/Jakobsen_Thomas/pubs/interpol.ps.gz

[17] C. Jutla. – Encryption modes with almost free message integrity. – *In : Advances in Cryptology, proceedings of Eurocrypt'01*, Innsbruck, Autriche, 7–10 mai 2001, éd. par B. Pfitzmann. – LNCS, vol. 2045, pp. 529–544. Springer-Verlag, 2001. Cf. http://csrc.nist.gov/encryption/modes/workshop1/

[18] L. Knudsen. – Truncated and higher order differentials. – *In : Fast Software Encryption : Second International Workshop*, Leuven, Belgique, 14–16 déc. 1994, éd. par B. Preneel. – LNCS, vol. 1008, pp. 196–211. Springer-Verlag, 1995.

[19] L. Knudsen, – Contemporary block ciphers. – *In : Lectures on Data Security. Modern Cryptology in in Theory and Practice,* éd. par I. Damgård. – LNCS Tutorial 1561, pp. 105–126. Springer-Verlag, 1999. Cf. http://www.ii.uib.no/~larsr/papers/survey98.ps.

[20] L. Knudsen, M. J. B. Robshaw et D. Wagner. – Truncated differentials and Skipjack. – *In : Advances in Cryptology, proceedings of Crypto'99*, Santa Barbara, CA, 15–19 août 1999, éd. par M. Wiener. – LNCS, vol. 1666, pp. 165–180. Springer-Verlag, 1999. Cf. http://www.cs.berkeley.edu/~daw/papers/skipjack-crypto99.ps.

[21] L. Knudsen et D. Wagner. – Integral Cryptanalysis. – *In : Advances in Cryptology, proceedings of FSE'02*, Leuven, Belgique, 4–6 février 2002, éd. par J. Daemen et V. Rijmen. – LNCS. Springer-Verlag, 2002. À paraître.

[22] X. Lai. – *On the design and security of block ciphers.* – ETH Series in Information Processing, vol. 1. Hartung-Gorre Verlag, Konstanz, Switzerland, 1992.

[23] X. Lai. – Higher order derivatives and differential cryptanalysis. – *In : Proc. Symposium on Communication, Coding and Cryptography*, éd. par R. E. Blahut, D. J. Costello, U. Maurer et T. Mittelholzer. – 1994.

[24] X. Lai, J. Massey et S. Murphy. – Markov ciphers and differential cryptanalysis. – *In : Advances in Cryptology, proceedings of Eurocrypt'91*, Brighton, UK, 8–11 avril

1991, éd. par D. Davies. – LNCS, vol. 547, pp. 17–38. Springer-Verlag, 1991. Cf. http://www.cs.rhbnc.ac.uk/~sean/xuejia.ps

[25] M. Luby et C. Rackoff. – *How to construct pseudorandom permutations from pseudorandom functions.* – SIAM Journal on Computing, vol. 17, n°2, pp. 373–386, avril 1988.

[26] M. Matsui. – Linear Cryptanalysis Method for DES Cipher. – *In : Advances in Cryptology, proceedings of Eurocrypt'93*, Lofthus, Norvège, 23–27 mai 1993, éd. par T. Helleseth. – LNCS, vol. 765, pp. 386–397. Springer-Verlag, 1994.

[27] NIST. – FIPS 46-3, Data Encryption Standard (DES), oct. 1999. Cf. http://csrc.nist.gov/encryption/tkencryption.html

[28] NIST. – FIPS 185, Escrowed Encryption Standard (EES), fév. 1994. Cf. http://csrc.nist.gov/encryption/tkencryption.html

[29] NIST. – FIPS 197, Advanced Encryption Standard (AES), nov. 2001. Cf. http://csrc.nist.gov/encryption/aes/

[30] NIST. – FIPS 81, DES Modes of Operation, déc. 1980. Cf. http://www.itl.nist.gov/fipspubs/fip81.htm

[31] NIST. – Modes of operation. Cf. http://csrc.nist.gov/encryption/modes/

[32] NIST. – Draft FIPS, HMAC, jan. 2001. Cf. http://csrc.nist.gov/encryption/hmac/

[33] K. Nyberg. – Differentially uniform mappings for cryptography. – *In : Advances in Cryptology, proceedings of Eurocrypt'93*, Lofthus, Norvège, 23–27 mai 1993, éd. par T. Helleseth. – LNCS, vol. 765, pp. 55–64. Springer-Verlag, 1994.

[34] K. Nyberg et L. Knudsen. – Provable Security Against a Differential Attack. *Journal of Cryptology*, vol. 8, n° 1, 1995, pp. 27–37.

[35] J. Patarin. – New results on pseudorandom permutation generators based on the DES scheme. – *In : Advances in Cryptology, proceedings of Crypto'91*, Santa Barbara, Californie, 11–15 août 1991, éd. par J. Feigenbaum. – LNCS, vol. 576, pp. 301–312. Springer-Verlag, 1991.

[36] J. Patarin. – About Feistel schemes with Six (or more) rounds. – *In : Fast Software Encryption : Fifth International Workshop*, Paris, France, 23–25 mars 1998, éd. par S. Vaudenay. – LNCS, vol. 1372, pp. 103–121. Springer-Verlag, 1998.

[37] J. Patarin. – Generic attacks on Feistel schemes. – *In : Advances in Cryptology, proceedings of Asiacrypt'01*, Gold Coast, Australie, 9–13 décembre 2001, éd. par C. Boyd – LNCS, vol. 2248, pp. 222–238. Springer-Verlag, 2001.

[38] P. Rogaway, M. Bellare, J. Black and T. Krovetz. – OCB mode : parallelizable authenticated encryption. Cf. http://eprint.iacr.org/2001/026/

[39] A. Shamir et A. Kipnis. – Cryptanalysis of the HFE public key cryptosystem. – *In : Advances in Cryptology, proceedings of Crypto'99*, éd. par M. Wiener – LNCS, vol. 1666, pp. 19–30. Springer-Verlag, 1999.

[40] V. Shoup. – A proposal for an ISO standard for public key encryption (version 2.0), sept. 2001. Cf. http://eprint.iacr.org/2001/112/.

[41] S. Vaudenay. – *Vers une théorie du chiffrement symétrique.* Habilitation à diriger des recherches, 1998.

[42] S. Vaudenay. – Resistance Against General Iterated Attacks. – *In : Advances in Cryptology, proceedings of Eurocrypt'99*, Prague, République Tchèque, 2–6 mai 1999, éd. par J. Stern. – LNCS, vol. 1592, pp. 255–271. Springer-Verlag, 1999. Cf. `http://lasecwww.epfl.ch/pub/lasec/doc/Vau99a.ps`

[43] D. Wagner. – The boomerang attack. – *In : Advances in Cryptology, proceedings of FSE'99*, Rome, Italie, 24–26 mars 1999, éd. par L. Knudsen – LNCS, vol. 1636. Springer-Verlag, 1999. Cf. `http://www.cs.berkeley.edu/~daw/papers/boomerang-fse99.ps`.

Chapitre 3 (6h)

Vingt-cinq ans d'attaques de RSA

— par Phong Nguyen

Sommaire

3.1 Le cryptosystème RSA 44
3.1.1 Description 44
3.1.2 Complexité 46
3.2 RSA et la factorisation d'entiers 46
3.3 Attaques élémentaires 47
3.3.1 Recherche exhaustive 47
3.3.2 Fuite d'information 47
3.3.3 Messages courts 48
3.3.4 Attaque "broadcast" 48
3.3.5 Attaque à chiffré choisi 48
3.3.6 Attaque par le milieu 49
3.4 Attaques sur les implémentations du RSA 50
3.4.1 Attaques par chronométrage 50
3.4.2 Attaques par erreur 51
3.4.3 L'attaque de Bleichenbacher sur PKCS 1 51
3.5 Attaques simples du RSA à petit exposant 51
3.5.1 Petit exposant public 52
3.5.2 Petit exposant privé 52
3.6 Attaques à base de géométrie des nombres 53
3.6.1 Géométrie des nombres 53
Définitions 53
Le théorème de Minkowski 56
Minima successifs 57
L'algorithme LLL 58
3.6.2 Le théorème de Coppersmith 59
3.6.3 Application au RSA à base d'identité 60
3.6.4 Application au RSA à petit exposant public 60
Chiffrement stéréotypé 60
Chiffrement multiple 61
Chiffrement avec petit aléa 61

Le cryptosystème RSA a été inventé il y a vingt-cinq ans. C'est aujourd'hui le cryptosystème asymétrique le plus utilisé de par le monde. Dans ce chapitre, nous présenterons les principales attaques découvertes contre RSA depuis son apparition. Si aucune de ces attaques n'est réellement dévastatrice, elles démontrent toutefois qu'il faut implémenter RSA avec beaucoup de précautions. Elles illustrent également la difficulté à définir des notions de sécurité convenables, en chiffrement asymétrique comme en signature. Ce chapitre s'inspire largement du panorama [2] dressé par Boneh.

3.1 Le cryptosystème RSA

Le chiffrement symétrique existe depuis la création de la cryptographie. Il ne fut remis en question qu'en 1976 par Diffie et Hellman.[1] Leur article [13] proposait de rompre la symétrie des rôles de l'expéditeur et du destinataire, en introduisant l'idée de *chiffrement à clef publique* (appelé aussi *chiffrement asymétrique*) selon laquelle seul le déchiffrement resterait confidentiel. Cela revient à rendre la clef de chiffrement publique (même pour l'adversaire), la clef de déchiffrement restant privée. Ces deux clefs sont en fait reliées mathématiquement (parce que la fonction de déchiffrement est l'inverse de la fonction de chiffrement), mais il doit être impossible en pratique de retrouver la clef privée à partir uniquement de la clef publique. La situation rappelle dans une certaine mesure l'exemple des cadenas que n'importe qui peut refermer une fois ouverts, mais que seul le propriétaire de la clef peut ouvrir. Les propriétés requises pour le cryptosystème sont plus fortes qu'en cryptographie symétrique, si bien que Diffie et Hellman ne purent illustrer par aucun exemple leur nouvelle théorie. Suite à la publication de l'article [13], trois chercheurs du M.I.T., nommés Rivest, Shamir et Adleman, tentèrent de prouver qu'un cryptosystème asymétrique ne pouvait exister. Mais après de nombreux essais infructueux, ils découvrirent au contraire un candidat possible de cryptosystème asymétrique [27], aujourd'hui connu sous le nom de RSA, et qui fut présenté pour la première fois dans le numéro d'août 1977 de *Scientific American*.

3.1.1 Description

Dans le système RSA, on commence par choisir aléatoirement deux nombres premiers [2] suffisamment grands p et q, de taille sensiblement égale, par exemple 512 bits. On calcule $n = pq$. On note mZ_n l'anneau $\mathbb{Z}/n\mathbb{Z}$ des classes d'équivalence modulo n. Le groupe multiplicatif $\mathbb{Z}_n^*$ des éléments inversibles a pour ordre $\varphi(n) = (p-1)(q-1)$. On choisit deux entiers d et e supérieurs à 3 et premiers avec $\varphi(n)$ tels que $ed \equiv 1 \pmod{\varphi(n)}$. Généralement, on choisit d'abord l'un des deux paramètres d ou e (dans $[3, \varphi(n)-1]$) et l'on en déduit l'autre. On recommande de choisir ces paramètres aléatoirement, mais il est courant de voir des implémentations industrielles utiliser un petit e pour des raisons d'efficacité, comme par exemple $e = 3$ ou $e = 2^{16} + 1$. On verra par la suite des attaques tirant parti de choix particuliers de d ou e. L'entier e est appelé *exposant de chiffrement* (d'où le choix de la

[1] Du moins, dans la recherche publique. De récentes révélations suggèrent en effet que cette remise en question intervint dès 1970 dans les services du chiffre britanniques (voir [8]). En tout état de cause, ces mêmes services ne semblent pas avoir mesuré l'importance de cette notion, contrairement à Diffie et Hellman.

[2] Nous n'aborderons pas ici la génération de nombres premiers. Voir par exemple [24].

lettre e pour *encryption*) et est rendu public, tandis que l'entier d est appelé *exposant de déchiffrement* (d'où le choix de d pour *decryption*) et est gardé secret. La clef publique RSA est $\mathsf{pk} = (n, e)$, et la clef privée RSA est $\mathsf{sk} = d$.

L'ensemble des messages est $\mathbb{Z}_n$. Un message $m \in \mathbb{Z}_n$ est chiffré en

$$c = m^e \bmod n.$$

Le déchiffrement d'un chiffré c n'est autre que :

$$c^d \bmod n.$$

Le déchiffrement fonctionne car :

$$\forall m \in \mathbb{Z}, m^{ed} \equiv m \pmod{n}.$$

Ce résultat se démontre simplement à l'aide des restes chinois et du petit théorème de Fermat, et est vrai pour tout module n sans facteur carré. Il suffit en effet de prouver cette identité modulo p et q, ce qui peut se déduire du petit théorème de Fermat qui assure que pour tout nombre premier p :

$$\forall m \in \mathbb{Z}_p^*, m^{p-1} \equiv 1 \pmod{p}.$$

Le petit théorème de Fermat peut se voir comme un cas particulier du théorème de Lagrange, puisque l'ordre de $\mathbb{Z}_p^*$ vaut $p - 1$ lorsque p est premier.

La description de RSA que l'on vient de donner est celle que l'on peut trouver dans l'article original [27] et dans de nombreux ouvrages de cryptographie, mais on sait aujourd'hui qu'il ne faut surtout pas utiliser RSA tel quel. En effet, cette version basique du cryptosystème RSA ne satisfait pas de fortes notions de sécurité. On verra par la suite les principaux problèmes de sécurité posés par cette version basique. Ce n'est que dans le chapitre suivant que seront définies formellement les notions de sécurité communément admises, et présentées les modifications de RSA pour satisfaire ces notions de sécurité.

Le cryptosystème RSA peut aussi s'utiliser en signature, et cette utilisation est peut-être encore plus répandue que celle du chiffrement RSA. On utilise les mêmes clefs $\mathsf{pk} = (n, e)$ et $\mathsf{sk} = d$ qu'en chiffrement. La signature d'un message $m \in \mathbb{Z}_n$ préalablement haché n'est autre que :

$$s = m^d \bmod n.$$

On vérifie la validité de la signature s en testant l'égalité :

$$m \equiv s^e \pmod{n}.$$

Mais comme en chiffrement, ce n'est pas cette description de base de la signature RSA qui est utilisée en pratique.

En fait, en termes cryptographiques, on n'a pour l'instant défini qu'une *fonction à sens unique à trappe*, et non un algorithme de chiffrement ou de signature au sens strict. La fonction en question est $m \mapsto m^e$. C'est une permutation de $\mathbb{Z}_n$ dont l'inverse est $m \mapsto m^d$. Cette fonction est donc facile à évaluer en connaissance de la clef publique $\mathsf{pk} = (n, e)$, mais *a priori*, on ne sait l'inverser (en moyenne) qu'en connaissance d'une trappe, à savoir la clef privée $\mathsf{sk} = d$, d'où le nom de fonction à sens unique à trappe. Il a été montré que l'on pouvait construire des algorithmes de chiffrement ou de signature sûrs à partir de n'importe quelle fonction à sens unique à trappe, et pas seulement la fonction RSA.

3.1.2 Complexité

Le chiffrement et le déchiffrement RSA nécessitent tous deux une exponentiation modulo n. Dans le cas général où les exposants d et e sont choisis aléatoirement, leur taille est donc celle de $\varphi(n)$, soit celle de n puisque $\varphi(n) = (p-1)(q-1)$. Avec les algorithmes usuels d'exponentiation (tels que celui décrit en section 1.4.1), le chiffrement et le déchiffrement coûtent donc tous deux $O(\log n)$ multiplications modulo n. Comme ces multiplications peuvent s'effectuer[3] en temps $O(\log^2 n)$, on obtient une complexité de chiffrement et de déchiffrement de $O(\log^3 n)$.

Cette complexité, bien que polynomiale, n'est pas négligeable, notamment pour des environnements à puissance de calcul réduite tels que la carte à puce. Aussi est-il tentant de choisir soit un petit e, soit un petit d, de façon à accélérer le chiffrement ou le déchiffrement. On évoquera plus tard les problèmes posés par ces choix particuliers d'exposants. Si d n'est pas petit, on peut légèrement accélérer le déchiffrement à l'aide des restes chinois. En effet, pour calculer $m^d \bmod n$, il suffit de calculer $m^d \bmod p$ et $m^d \bmod q$, puis d'appliquer les restes chinois dont le coût est négligeable par rapport au calcul de $m^d \bmod n$ si d est grand. Or $m^d \equiv m^{d \bmod (p-1)} \pmod p$ et $m^d \equiv m^{d \bmod (q-1)} \pmod q$, de sorte que si l'on conserve les valeurs de p et q, on peut ramener le calcul de $m^d \bmod n$ à deux exponentielles d'exposant de taille divisée par deux, avec un module de taille également divisée par deux. Comme la complexité d'une exponentielle est de $O(\log^3 n)$, on divise ainsi approximativement le temps de calcul global par quatre : seule la constante de la complexité est améliorée.

3.2 RSA et la factorisation d'entiers

Les liens entre RSA et le problème de la factorisation d'entiers peuvent se résumer par le résultat suivant :

Théorème 3.1 *Soient* $\mathsf{pk} = (n, e)$ *et* $\mathsf{sk} = d$ *des clefs RSA, avec* $n = pq$*. Si l'on connaît* pk*,* p *et* q*, alors on peut calculer* sk *en temps polynomial. Réciproquement, si l'on connaît* pk *et* sk*, alors on peut calculer* p *et* q *en temps polynomial probabiliste.*

Le sens direct est évident puisque $\varphi(n) = (p-1)(q-1)$. Ainsi, si l'on sait factoriser efficacement, on peut casser RSA, et de fait, la factorisation est dans le cas général la meilleure attaque connue contre RSA. Nous n'aborderons pas ici le problème fascinant qu'est la factorisation (voir par exemple [9]), mais précisons juste que le meilleur algorithme connu à ce jour est le crible algébrique [21], dont la complexité pour factoriser un nombre n est heuristiquement de :

$$\exp\left(O(1)(\log n)^{1/3}(\log\log n)^{2/3}\right).$$

Le crible algébrique détient le record actuel de factorisation pour un nombre RSA, qui concerne un nombre de 524 bits faisant 158 chiffres décimaux, factorisé en janvier 2002.

La réciproque est plus difficile à démontrer. Tout d'abord, on remarque par les restes chinois qu'il y a exactement quatre racines carrées de l'unité dans $\mathbb{Z}_n^*$: deux triviales (1 et -1) et deux autres qui permettent de calculer p et q en temps polynomial (si $x^2 \equiv 1 \pmod n$ avec $x \neq 1$, alors le pgcd de $x-1$ avec n vaut soit p soit q). On peut donc déterminer p

[3] La complexité que l'on donne n'est pas optimale, puisque l'on peut l'améliorer avec l'astuce de Karatsuba ou la transformée de Fourier discrète, mais elle correspond mieux aux implémentations usuelles.

et q en temps polynomial probabiliste si l'on arrive à engendrer aléatoirement des racines carrées de l'unité.

Considérons à présent l'exposant $\lambda(n)$ de $\mathbb{Z}_n^*$, qui est par définition le plus grand des ordres de tous les éléments de $\mathbb{Z}_n^*$, soit encore le plus petit entier λ non nul tel que pour tout entier m de $\mathbb{Z}_n^*$, $m^\lambda \equiv 1 \pmod n$. Le théorème de Lagrange implique que $\lambda(n)$ divise $\varphi(n)$, ce que l'on peut voir aussi parce que $\lambda(n)$ est le ppcm de $p-1$ et $q-1$. Comme $ed \equiv 1 \pmod{\varphi(n)}$ et donc $ed \equiv 1 \pmod{\lambda(n)}$, on peut déduire de pk et sk un multiple non nul de $\lambda(n)$, à savoir $ed-1$. Ce multiple peut s'écrire sous la forme $m = 2^s t$ avec t impair. Alors l'un des nombres $m, m/2, \cdots, m/2^s$ est de la forme $k\lambda(n)$ avec k impair, et ces nombres sont en nombre polynomial en $\log n$. Considérons alors le morphisme de groupes μ qui à tout m de $\mathbb{Z}_n^*$ associe $m^{\lambda(n)k/2} \bmod n$. L'image de μ est un sous-groupe du groupe d'ordre 4 des racines carrées de l'unité. En regardant modulo p et q, on voit que cette image est soit le groupe des racines tout entier, soit un sous-groupe d'ordre 2 non trivial, par définition de l'exposant. Dans les deux cas, si l'on tire m uniformément dans $\mathbb{Z}_n^*$, il y a une chance sur deux pour que $\mu(m)$ soit une racine carrée non triviale, et donc que l'on factorise n, ce qui achève la démonstration du théorème 3.1. Au passage, remarquons que ce résultat montre que deux personnes ne doivent jamais partager le même module RSA, sous peine de partager leur clef privée.

Mais le théorème 3.1 ne signifie pas que la sécurité de RSA est équivalente à la difficulté de la factorisation. En effet, rien ne prouve que la connaissance de d soit indispensable au calcul de la fonction $m \mapsto m^d \bmod n$, c'est-à-dire à l'inversion de la fonction $m \mapsto m^e \bmod n$. Certains [7] suggèrent même que le problème d'extraction des racines e-ièmes modulo n serait plus facile que la factorisation de n lorsque e est petit. En ce sens, l'équivalence entre la factorisation et l'inversion de RSA est un problème ouvert.

3.3 Attaques élémentaires

3.3.1 Recherche exhaustive

On remarque que la fonction de chiffrement RSA $m \mapsto m^e \bmod n$ est déterministe : un message est toujours chiffré en le même chiffré. Étant donné un chiffré c et un message m, on peut donc vérifier si c est le chiffrement de m. Ainsi, si l'ensemble des messages possibles est connu et de petite taille, alors on peut décrypter par recherche exhaustive, en testant tous les messages possibles. Par exemple, si l'on chiffre un texte clair en chiffrant au moyen de RSA chaque lettre du texte séparément, alors n'importe qui pourra décrypter.

Pour éviter la recherche exhaustive, il est indispensable de randomiser les messages avant chiffrement. On ne peut atteindre de bonnes notions de chiffrement qu'en ayant une fonction de chiffrement probabiliste (en rajoutant de l'aléa à chaque chiffrement), et non déterministe. C'est pourquoi une fonction à sens unique à trappe doit être transformée avant d'être utilisée en chiffrement asymétrique.

3.3.2 Fuite d'information

On remarque qu'un chiffré RSA révèle au moins un bit d'information sur le message clair. En effet si $c = m^e \bmod n$, on a, comme e est nécessairement impair :

$$\left(\frac{c}{n}\right) = \left(\frac{m}{n}\right)^e = \left(\frac{m}{n}\right).$$

Le symbole de Jacobi du message m peut donc se déduire du chiffré c. Une notion de sécurité forte doit empêcher de déduire facilement du chiffré toute information sur le message.

3.3.3 Messages courts

Supposons que l'on chiffre un message court m. Le chiffré de m est $c = m^e \bmod n$. Si m est tellement court que $m < n^{1/e}$ (ce qui ne peut arriver que si e est bien plus petit que $\varphi(n)$, par exemple si $e = 3$), alors $c = m^e$ dans $\mathbb{Z}$, et donc on peut efficacement décrypter c en extrayant une racine e-ième dans $\mathbb{Z}$. Ainsi, il n'y a absolument aucune sécurité si l'on chiffre une clef de session 128 bits au moyen de RSA-1024 d'exposant 3 avec la fonction de chiffrement de base. Là encore, cet exemple montre qu'il faut randomiser les messages.

3.3.4 Attaque "broadcast"

L'attaque précédente sur les messages courts peut se généraliser au scénario du *broadcast* où un même utilisateur envoie des messages chiffrés à plusieurs personnes disposant chacune de leur propre clef publique. Supposons donc qu'Alice communique avec r personnes disposant chacune d'une clef publique RSA $\mathsf{pk}_i = (n_i, e)$ distincte, mais utilisant le même petit exposant public e. Supposons en outre qu'Alice envoie un même message m à ces r personnes, mais en chiffrant avec la clef respective de ces r personnes. Ainsi, Alice envoie en fait $m^e \bmod n_1, m^e \bmod n_2, \ldots, m^e \bmod n_r$. Si les modules n_i ne sont pas premiers entre eux, alors on peut casser certaines des clefs RSA par simple pgcd. Sinon, on peut par restes chinois calculer $m^e \bmod \prod_{i=1}^{r} n_i$. Mais si le nombre r de personnes est supérieur à e, alors on connaît exactement m^e dans $\mathbb{Z}$, puisque m est inférieur à chacun des n_i. On peut ainsi décrypter en extrayant une racine e-ième dans $\mathbb{Z}$.

Pour contrecarrer cette attaque (remarquée pour la première fois par Hastad [18]), il faut transformer le message m avant de le chiffrer. Hastad a montré dans [18] que certaines transformations naturelles n'étaient pas sûres, résultat amélioré plus tard par Coppersmith [10]. Ces attaques bien plus complexes, à base de géométrie des nombres, seront présentées à la fin de ce chapitre.

3.3.5 Attaque à chiffré choisi

On remarque que la fonction de chiffrement RSA est *multiplicative*, car l'exponentiation est un morphisme de groupes. En effet, si m_1 et m_2 sont deux messages dans $\mathbb{Z}_n$, chiffrés respectivement en $c_1 = m_1^e \bmod n$ et $c_2 = m_2^e \bmod n$, alors :

$$c_1 c_2 \equiv (m_1 m_2)^e \pmod{n}.$$

Ainsi, le produit de deux chiffrés n'est autre que le chiffré du produit. Cette propriété induit ce que l'on appelle une *attaque à chiffré choisi* : supposons qu'un attaquant veuille décrypter un chiffré c, et qu'il puisse faire appel à un oracle de déchiffrement mais par sur l'instance c, c'est-à-dire qu'il peut obtenir le déchiffrement de tout chiffré sauf c. On obtient une attaque à chiffré choisi contre RSA :

- L'attaquant choisit un élément c_1 uniformément au hasard dans $\mathbb{Z}_n^*$.
- Il calcule $c_2 = c \cdot c_1 \bmod n$.
- Il demande le déchiffrement de c_1 et c_2, notés m_1 et m_2. Avec probabilité écrasante, c_1 et c_2 sont bien distincts de c.
- Par multiplicativité de RSA, c se déchiffre en $m_2 m_1^{-1} \bmod n$.

Cette attaque à chiffré choisi peut sembler artificielle, mais on admet aujourd'hui qu'une bonne notion de sécurité doit éviter ce genre d'attaques (voir le chapitre suivant). Cette attaque peut facilement s'adapter au cas de la signature : on suppose que Bob dispose de clefs RSA, et qu'un attaquant veut obtenir la signature de Bob sur un certain message m. L'allure de m étant douteuse, Bob refuse de signer m. Mais comme précédemment, l'attaquant peut facilement engendrer deux messages m_1 et m_2 d'apparence plus aléatoire, et demander à Bob les signatures s_1 et s_2 de m_1 et m_2. Si m_1 et m_2 sont choisis convenablement, l'attaquant pourra facilement obtenir la signature $s = m^d \bmod n$ à partir de s_1 et s_2. Dans la terminologie des signatures, on appelle une telle attaque une *attaque à message choisi*. Cet exemple montre la nécessité d'appliquer préalablement à la signature une fonction de hachage. Dans le cas de RSA, une fonction de hachage appropriée permettra de masquer la multiplicativité.

3.3.6 Attaque par le milieu

La multiplicativité de RSA donne lieu à ce que l'on appelle communément une *attaque par le milieu* (voir [6]). Supposons que l'on chiffre un message court $m : 0 \leq m < M$ avec M petit. Soit $c = m^e \bmod n$ le chiffré correspondant. La recherche exhaustive montre qu'à partir de c et n, on peut retrouver n en effectuant $O(M)$ exponentiations modulaires (c'est une recherche exhaustive). Ainsi, on pourrait croire que chiffrer directement une clef de session de 64 bits ne pose aucun problème. L'attaque par le milieu suivante montre qu'il n'en est rien.

Supposons que m puisse s'écrire sous la forme $m = m_1 \cdot m_2$ avec $0 \leq m_1, m_2 < M^{1/2}$. Alors par multiplicativité de RSA :

$$\frac{c}{m_2^e} \equiv m_1^e \pmod{n}.$$

On en déduit une attaque par le milieu :

- On construit la liste L de tous les $m_1^e \bmod n$ avec $0 \leq m_1 < M^{1/2}$.
- Si l'on veut décrypter c, on calcule pour tout m_2 tel que $0 \leq m_2 < M^{1/2}$, la valeur $c/m_2^e \bmod n$, et l'on teste si cette valeur appartient à L : si oui, on obtient ainsi les valeurs de m_1 et m_2 et donc celle de m.

Le coût de cette attaque est essentiellement de $O(M^{1/2})$ exponentiations modulo n, et il faut stocker et trier $O(M^{1/2})$ éléments de $\mathbb{Z}_n$ (en fait, on peut se contenter de suffisamment de bits de poids fort). Par rapport à la recherche exhaustive, on a gagné en temps ce que l'on a perdu en mémoire, mais cette complexité devient par exemple très réaliste dans le cas où $M = 2^{64}$.

Il reste à évaluer la probabilité que m puisse s'écrire sous la forme $m = m_1 \cdot m_2$ avec $0 \leq m_1, m_2 < M^{1/2}$. Pour plus de détails, voir [6]. Soit $P_\alpha(M)$ la probabilité qu'un nombre choisi uniformément au hasard dans $\{0, \ldots, M\}$ puisse s'écrire $m_1 \cdot m_2$ avec $m_i \leq M^\alpha$. Erdös a démontré que $P_{1/2}(M)$ convergeait très lentement vers 0 lorsque M tendait vers l'infini. Cependant, pour des valeurs usuelles, cette probabilité est non négligeable : par exemple, $P_{1/2}(2^{40})$ et $P_{1/2}(2^{64})$ sont proches de $1/5$. En outre, on sait que si $\alpha > 1/2$, $P_\alpha(M) \geq \ln(2\alpha)$. Ainsi, une proportion significative des messages de $\{0, \ldots, M\}$ peut s'attaquer par l'attaque par le milieu précédente, ce qui démontre que la sécurité du chiffrement RSA d'un message de $\{0, \ldots, M\}$ est au mieux en $O(M^{1/2})$.

3.4 Attaques sur les implémentations du RSA

Avant de voir des attaques plus élaborées du RSA, on va d'abord s'intéresser aux attaques concernant les implémentations du RSA. Ces attaques montrent qu'une implémentation du RSA ne doit pas seulement se protéger des attaques mathématiques du chiffrement RSA.

3.4.1 Attaques par chronométrage

Considérons une carte à puce contenant une clef privée RSA. Normalement, un attaquant mettant la main sur la carte ne peut arriver à déterminer cette clef privée, en raison de protections physiques. Kocher [20] a néanmoins démontré qu'en mesurant précisément le temps pris par la carte pour effectuer un déchiffrement RSA (ou une signature), un attaquant pouvait rapidement déterminer l'exposant privé d pour les implémentations usuelles du RSA.

Supposons par exemple que la carte implémente l'algorithme usuel d'exponentiation modulaire. Soit $d = d_r d_{r-1} \cdots d_0$ la décomposition binaire de d : $d = \sum_{i=0}^{r} d_i 2^i$. L'algorithme usuel pour calculer $m^d \bmod n$ est le suivant :

1. On pose $a \longleftarrow m$ et $b \longleftarrow 1$.
2. Pour i allant de 0 à r :
3. (a) Si $d_i = 1$, faire $b \longleftarrow b \cdot a \bmod n$.
   (b) $a \longleftarrow a^2 \bmod n$.
4. Renvoyer b, qui vaut $m^d \bmod n$.

L'attaquant Albert demande à la carte de signer un grand nombre de messages aléatoires $m_1, \ldots, m_k \in \mathbb{Z}_n^*$ et mesure le temps T_i pris par la carte pour signer chacun des messages. Albert tente de retrouver tous les bits de d en commençant par le bit de poids faible. Par définition des clefs RSA, on sait déjà que $d_0 = 1$. Initialement, $a = m^2 \bmod n$ et $b = m$. Si $d_1 = 1$, la carte calcule $b \cdot a = m \cdot m^2 \bmod n$. Soit t_i le temps pris par la carte pour calculer $m_i \cdot m_i^2 \bmod n$. Les t_i sont distincts puisque ce temps dépend de la valeur de m_i. Ces temps t_i sont mesurés avant de mener l'attaque, une fois qu'Albert connaît les spécifications physiques de la carte.

Kocher a remarqué que lorsque $d_1 = 1$, les deux ensembles $\{t_i\}$ et $\{T_i\}$ étaient corrélés. Par exemple, si pour un i, t_i est bien plus grand que son espérance, alors T_i est aussi bien plus grand que son espérance. Mais si $d_1 = 0$, les deux ensembles se comportent comme des variables aléatoires indépendantes. En mesurant la corrélation, Albert peut ainsi déterminer si d_1 vaut 0 ou 1. On détermine les bits suivants de la même façon.

Il n'est pas difficile de se prémunir contre ce genre d'attaques, par exemple en modifiant l'algorithme pour uniformiser le temps de calcul, ou en "aveuglant" les messages (la carte choisit au hasard $u \in \mathbb{Z}_n^*$, et calcule la signature de $m \cdot u^e \bmod n$ au lieu de m).

Kocher a récemment découvert un autre type d'attaques, en mesurant cette fois la consommation de la carte. En effet, la consommation de la carte permet facilement de déterminer si la carte effectue une ou deux multiplications, révélant ainsi les bits de d. Ces attaques semblent plus efficaces que les attaques par chronométrage, mais on peut également s'en protéger.

3.4.2 Attaques par erreur

On a vu dans la section 3.1.2 qu'il était avantageux d'utiliser les restes chinois pour accélérer le calcul de $m^d \bmod n$. Boneh, DeMillo et Lipton [3] ont remarqué que cette méthode présentait le danger potentiel suivant. Supposons que par accident (ou par intervention d'un adversaire), la carte fasse une seule erreur d'instruction lors du calcul de la signature, de sorte que seule la valeur de $m^d \bmod p$ soit correctement calculée, et non celle de $m^d \bmod q$. La signature incorrecte s vérifiera donc $s^e \equiv m \pmod p$ et $s^e \not\equiv m \pmod q$, et donc le pgcd de $s^e - m$ et n révélera la factorisation de n.

3.4.3 L'attaque de Bleichenbacher sur PKCS 1

Cette attaque a eu beaucoup de retentissement lorsqu'elle a été publiée en 1998 par Bleichenbacher [1].

On a vu en section 3.3 plusieurs attaques très simples qui montrent la nécessité de formater les messages avant chiffrement RSA, en rajoutant notamment des bits aléatoires. La norme PKCS proposée par les laboratoires RSA (voir [28]) présente une telle procédure de formatage, et c'est sans doute la norme la plus utilisée dans le monde pour le chiffrement RSA. Jusqu'à la publication de [1], on utilisait la version 1 de la norme PKCS, qui formatait un message m (de longueur inférieure à celle de n), de la façon suivante :

02	Aléa	00	m

Le premier bloc contenant "02" fait 16 bits (2 octets) et sert à indiquer que la norme PKCS 1 est utilisée et donc que l'on rajoute de l'aléa.

Lorsqu'un chiffré PKCS 1 est reçu par une machine, l'application concernée (par exemple un navigateur) le déchiffre, vérifie le premier bloc, et retire les octets d'aléa. Il s'avère que lorsque le bloc initial "02" est absent, certaines applications (notamment des vieilles implémentations de SSL) renvoient un message d'erreur "chiffré invalide". Bleichenbacher a démontré que ce message d'erreur avait des conséquences désastreuses : un attaquant recevant de tels messages d'erreur peut décrypter des chiffrés de son choix.

Supposons qu'Albert intercepte un chiffré c destiné a Bernard, et qu'il veuille le décrypter. Albert choisit un r aléatoire dans $\mathbb{Z}_n^*$, et calcule $c' = r \cdot c \bmod n$. Il envoie c' à la machine de Bob, qui tente de le déchiffrer, et indique ainsi à Albert si c' est un chiffré PKCS 1 valide. Albert apprend donc si oui ou non les 16 bits de poids fort du déchiffrement de c' sont égaux à 02. En d'autres termes, Albert dispose d'un oracle lui indiquant si les 16 bits de poids fort du déchiffrement de $r \cdot c \bmod n$ sont égaux à 02, pour tout r de son choix. Bleichenbacher a démontré qu'un tel oracle permettait de décrypter c en totalité.

Ce résultat n'est pas sans rapport avec ce que l'on appelle la sécurité bit à bit de RSA : on sait depuis une quinzaine d'années que chaque bit individuel de $m^d \bmod n$ est aussi difficile à calculer que la totalité de $m \bmod n$. Plus précisément, un oracle déterminant un bit (de position fixe) de $m^d \bmod n$ pour n'importe quel m permet de calculer la totalité de $m^d \bmod n$ pour tout m. Ce résultat subsiste même si l'oracle commet des erreurs, du moment que l'avantage (par rapport à un bit choisi au hasard) est non négligeable.

3.5 Attaques simples du RSA à petit exposant

Les attaques plus élaborées seront vues dans la prochaine section. Elles améliorent les attaques présentées ici.

3.5.1 Petit exposant public

On suppose ici que l'exposant public e est petit. On a déjà vu dans les sections 3.3.3 et 3.3.4 des attaques simples concernant ce choix particulier d'exposant. En voici une autre, décrite dans [11].

Supposons que l'on chiffre des messages reliés linéairement entre eux, par exemple si l'un des messages est le décalage de l'autre. Soient donc $m_1, m_2 \in \mathbb{Z}_n^*$ deux messages tels que $m_1 = am_2 + b \bmod n$ où a et b sont des entiers connus. Ces messages sont chiffrés en $c_1 = m_1^e \bmod n$ et $c_2 = m_2^e \bmod n$. [11] ont montré qu'un attaquant disposant de c_1, c_2, a, b et n, pouvait facilement retrouver m_1 et m_2.

En effet, posons $f(x) = ax + b$ de sorte que $m_1 = f(m_2) \bmod n$. On sait que m_2 est une racine du polynôme $g_1(x) = f(x)^e - c_1 \in \mathbb{Z}_n[x]$. De même, m_2 est une racine de $g_2(x) = x^e - c_2 \in \mathbb{Z}_n[x]$. Le facteur linéaire $x - m_2$ divise donc $g_1(x)$ et $g_2(x)$. Un attaquant va naturellement calculer le pgcd de $g_1(x)$ et $g_2(x)$, en utilisant l'algorithme d'Euclide : le fait que l'anneau $\mathbb{Z}_n[x]$ ne soit pas euclidien n'a pas d'importance puisque si jamais on a un problème avec des diviseurs de zéros, on aura factorisé n. Si le pgcd est linéaire, m_2 est donc découvert.

Il est facile de démontrer que le pgcd est toujours linéaire si $e = 3$. Pour $e > 3$, il s'avère qu'il l'est presque toujours. En outre, le calcul du pgcd nécessite un temps quadratique en e et $\log n$, et est donc prohibitif dès que e n'est pas très petit.

3.5.2 Petit exposant privé

La première attaque significative contre RSA à petit exposant privé est due à Wiener [30]. C'est une attaque astucieuse reposant sur la théorie des fractions continues (voir par exemple [17]) :

Théorème 3.2 (Wiener) *Soient $n = pq$ avec $q < p < 2q$, et $d < \frac{1}{3}N^{1/4}$. Étant donnés n et e tels que $ed \equiv 1 \ (mod\, \varphi(n))$, on peut calculer d en temps polynomial en $(\log n, \log e)$.*

Comme $ed \equiv 1 \pmod{\varphi(n)}$, il existe un petit entier k tel que $ed - k\varphi(n) = 1$. On remarque que k et d sont premiers entre eux, et que l'on a :

$$\left| \frac{e}{\varphi(n)} - \frac{k}{d} \right| = \frac{1}{d\varphi(n)}.$$

Ainsi, la fraction inconnue k/d est une très bonne approximation du rationnel $e/\varphi(n)$, lui-même très proche du rationnel e/n qui est cette fois connu. En effet, $\varphi(n) = n - p - q + 1$ d'où $|n - \varphi(n)| < 3\sqrt{n}$. Un calcul élémentaire montre alors que :

$$\left| \frac{e}{n} - \frac{k}{d} \right| \leq \frac{1}{dn^{1/4}} < \frac{1}{2d^2}.$$

Une telle inégalité est classique en approximation diophantienne. Le nombre de fractions (irréductibles) k/d avec $d < n$ telles que $|e/n - k/d| < 1/(2d^2)$ est inférieur à $\log_2 n$, et toutes ces fractions peuvent s'obtenir en temps polynomial au moyen du développement en fractions continues (voir [17, Th. 177]). Et l'on peut aisément savoir laquelle de ces fractions est la bonne, puisque d permet de factoriser n.

Plusieurs contre-mesures ont été proposées pour contrecarrer l'attaque de Wiener :

– Augmenter (artificiellement) la taille de e, en rajoutant un multiple approprié de $\varphi(n)$. Cela affaiblit la qualité de l'approximation par k/d, en augmentant la taille de k. Si le nouvel exposant public $e > n^{1.5}$, l'attaque de Wiener ne fonctionne plus, quelle que soit la taille de d.
– Utiliser les restes chinois. Au lieu de choisir un d petit, on choisit un d tel que $d \bmod p$ et $d \bmod q$ sont simultanément petits. Dans ce cas, la meilleure attaque connue dans ce cas est une attaque par le milieu nécessitant essentiellement un temps de $O(\min(\sqrt{d \bmod p}, \sqrt{d \bmod q}))$.
– Déséquilibrer la taille des facteurs, en choisissant un p beaucoup plus petit que q, de façon à augmenter la taille de $p + q$, et donc affaiblir la qualité de l'approximation diophantienne. Mais cette méthode a récemment été attaquée dans [14].
– Augmenter le nombre de facteurs de n. En choisissant par exemple $n = pqr$ avec p, q et r premiers de même taille, l'exposant 0.25 diminue un peu.

3.6 Attaques à base de géométrie des nombres

La géométrie des nombres est une branche de la théorie des nombres fondée il y a un peu plus d'un siècle par Minkowski comme un pont entre la géométrie, la théorie des formes quadratiques et l'approximation diophantienne. Elle étudie les réseaux, qui sont des arrangements réguliers (et infinis) de points de l'espace à n dimensions, apparus au dix-neuvième siècle à la fois en cristallographie et en arithmétique. Ses aspects algorithmiques ont acquis beaucoup d'importance à la fois en mathématiques et en informatique depuis la découverte, il y a vingt ans exactement, d'un algorithme efficace dit de réduction de réseaux, connu aujourd'hui sous le nom de LLL. Cet algorithme peut être vu comme une généralisation géométrique élégante et pertinente de l'algorithme d'Euclide du pgcd de deux entiers.

La géométrie des nombres algorithmique a eu des applications spectaculaires en cryptologie (voir le panorama [26]), notamment en cryptanalyse. L'algorithme LLL et ses variantes constituent à l'heure actuelle l'outil le plus populaire pour attaquer des cryptosystèmes asymétriques. En particulier, ils jouent un rôle essentiel dans les meilleures attaques connues contre RSA, factorisation comprise.

3.6.1 Géométrie des nombres

Il n'existe pas encore d'ouvrage satisfaisant couvrant à la fois les aspects mathématiques et algorithmiques de la géométrie des nombres : pour le point de vue mathématique, on pourra lire [23, 29, 16] ; tandis que [15], et dans une moindre mesure [9], adoptent un point de vue algorithmique. Une longue liste de références est disponible dans [25, Chapitre22] et [26].

Définitions

Dans tout ce qui suit, on considère $\mathbb{R}^n$ muni de sa structure naturelle d'espace euclidien, sans faire de distinction entre points et vecteurs, et l'on note μ la mesure de Lebesgue. On a défini informellement un réseau comme un arrangement régulier de points de l'espace. Plus précisément, on appelle *réseau* de $\mathbb{R}^n$ tout sous-groupe discret de $(\mathbb{R}^n, +)$, c'est-à-dire tout ensemble non vide stable par soustraction, pour lequel il existe une boule centrée en l'origine dont l'intersection avec l'ensemble se réduise à l'origine. C'est notamment le cas de

$\mathbb{Z}^n$. On s'intéressera d'ailleurs souvent aux *réseaux entiers*, *i.e.* les réseaux inclus dans $\mathbb{Z}^n$, soit encore les sous-groupes de $\mathbb{Z}^n$. Le théorème suivant caractérise les réseaux, et justifie notre interprétation :

Théorème 3.3 *Soit L une partie non vide de $\mathbb{R}^n$. Les propriétés suivantes sont équivalentes :*

1. *L est un réseau.*
2. *Il existe des vecteurs $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_d$ linéairements indépendants de $\mathbb{R}^n$ tels que L soit exactement l'ensemble $L(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ des combinaisons linéaires à coefficients entiers des $\mathbf{b}_i$, c'est-à-dire :*
$$L = \mathbb{Z}.\mathbf{b}_1 \oplus \mathbb{Z}.\mathbf{b}_2 \oplus \cdots \oplus \mathbb{Z}.\mathbf{b}_d.$$

Ainsi, les réseaux sont en quelque sorte des analogues discrets des espaces vectoriels : tout réseau est en particulier un $\mathbb{Z}$-module libre de rang fini. On appelle *base* d'un réseau L toute $\mathbb{Z}$-base de L en tant que $\mathbb{Z}$-module, ou de façon équivalente, toute famille de vecteurs satisfaisant la condition 2 du théorème précédent. Toutes les bases ont le même nombre d'éléments, égal à la dimension de $\mathbb{R}.L$, le sous-espace vectoriel engendré par L. Ce nombre est appelé le *rang*, ou la *dimension* de L, noté $\dim L$.

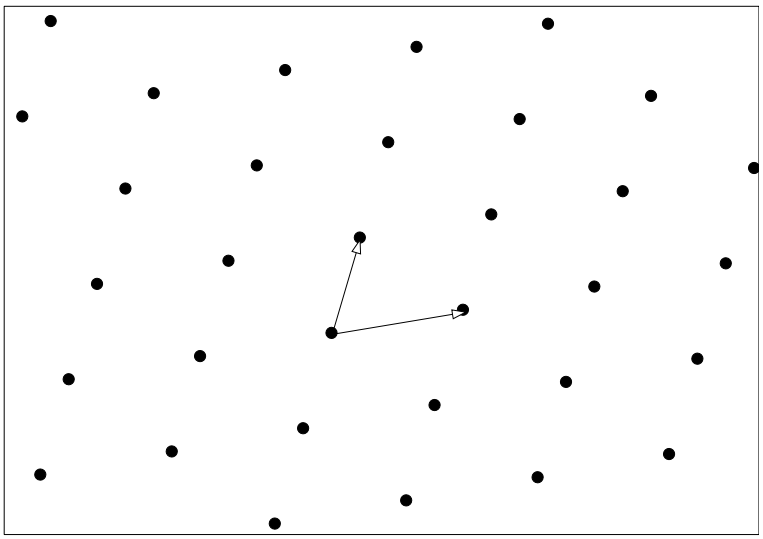


FIG. 3.1 – Un réseau de dimension 2, et une base.

Les bases diffèrent entre elles par des matrices de passages unimodulaires : si $\mathcal{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_d)$ est une base de L, alors une famille $(\mathbf{c}_1, \ldots, \mathbf{c}_d)$ de $\mathbb{R}^n$ est une base de L si et seulement si la matrice carrée $d \times d$ exprimant les $\mathbf{c}_i$ selon les $\mathbf{b}_i$ est à coefficients entiers, et de déterminant ± 1. En particulier, le déterminant $\Delta(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ de la matrice de Gram $(\langle \mathbf{b}_i, \mathbf{b}_j \rangle)_{1 \leq i,j \leq d}$ est un réel strictement positif indépendant de la base $\mathcal{B}$: on l'appelle le *discriminant* de L, noté $\Delta(L)$. Le *déterminant* ou *volume* de L est par définition la racine carrée du discriminant :

$$\det(L) = \sqrt{\Delta(L)}.$$

Remarquons que l'inégalité d'Hadamard nous assure que :

$$\det(L) \leq \prod_{i=1}^{d} \|\mathbf{b}_i\|.$$

L'appellation de volume se justifie facilement dans le cas important des réseaux *totaux* de $\mathbb{R}^n$, c'est-à-dire les réseaux dont la dimension est maximale, égale à celle de l'espace n. En effet, le déterminant du réseau n'est alors autre que le volume au sens usuel (la mesure de Lebesgue) des *briques élémentaires* du réseau, c'est-à-dire les ensembles de la forme

$$\left\{ \sum_{i=1}^{n} x_i \mathbf{b}_i \ : \ (x_1, \ldots, x_n) \in [0,1[^n \right\},$$

où les $\mathbf{b}_i$ forment une base du réseau (voir figure 3.2). Une telle brique est un cas particulier

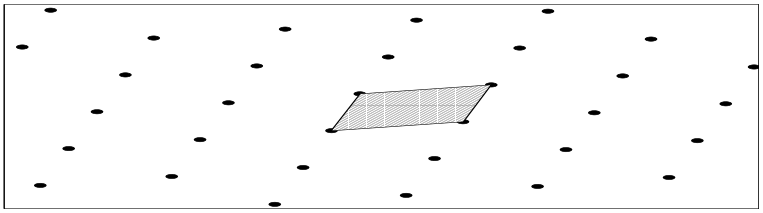


FIG. 3.2 – Interprétation géométrique du déterminant.

de *domaine fondamental* du réseau L, *i.e.* une partie D mesurable de $\mathbb{R}^n$ telle que les ensembles $\mathbf{b} + D$ forment, pour $\mathbf{b}$ décrivant L, un recouvrement de $\mathbb{R}^n$, et sont d'intérieurs deux à deux disjoints. Tous les domaines fondamentaux ont même volume, égal au volume du réseau. Un autre exemple de domaine fondamental est le *polyèdre de Voronoï*, qui est l'ensemble des points de l'espace pour lesquels l'origine est le point du réseau le plus proche.

La théorie des réseaux est souvent présentée dans la littérature en termes de réseaux totaux. Lorsqu'on s'en tient à un point de vue géométrique, cela ne pose pas de problèmes. En effet, pour tout réseau de dimension d dans $\mathbb{R}^n$, il existe une isométrie qui envoie l'espace engendré par le réseau dans $\mathbb{R}^d$. Et cette isométrie conserve les invariants géométriques (notamment, les volumes, les déterminants, les produits scalaires, ou les longueurs des vecteurs). Mais cette isométrie ne conserve pas, en général, le caractère entier des coordonnées des vecteurs. Aussi, lorsqu'on adopte un point de vue algorithmique ou combinatoire, on

ne peut pas se restreindre au cas des réseaux totaux. Cette approche privilégie en outre l'aspect « réseau » par rapport à l'aspect « formes quadratiques » (que nous occulterons). La théorie des réseaux fut historiquement étudiée en termes de formes quadratiques, toute base $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ d'un réseau de $\mathbb{R}^n$ définissant naturellement une forme quadratique définie positive sur $\mathbb{R}^d$ par :

$$q(x_1, \ldots, x_d) = \|x_1\mathbf{b}_1 + \cdots + x_d\mathbf{b}_d\|^2.$$

Le théorème de Minkowski

On peut remarquer que l'intersection d'un réseau avec un ensemble borné de l'espace est nécessairement finie. Le théorème fondamental de Minkowski donne une condition suffisante sur l'ensemble pour que cette intersection soit non triviale. Il peut s'interpréter comme une généralisation du célèbre lemme combinatoire des tiroirs (parfois appelé principe de Dirichlet), qui assure que lorsqu'on range plus de $n+1$ objets dans n tiroirs, au moins deux objets sont dans le même tiroir. Cette analogie est apparente dans le résultat suivant :

Lemme 3.1 (Blichfeldt) *Soient L un réseau total de $\mathbb{R}^n$, et F une partie mesurable de $\mathbb{R}^n$ telle que $\mu(F) > \det(L)$. Alors F contient deux vecteurs distincts dont la différence est dans L.*

Ce lemme est au cœur du théorème de Minkowski :

Théorème 3.4 (Minkowski) *Soit L un réseau total de $\mathbb{R}^n$. Soit C une partie mesurable de $\mathbb{R}^n$, convexe, symétrique par rapport à 0, et telle que*

$$\mu(C) > 2^n \det(L).$$

Alors C contient au moins un point non nul de L.

On remarque que la borne sur les volumes est la meilleure possible, en considérant

$$C = \left\{ \sum_{i=1}^{n} x_i\mathbf{b}_i \mid (x_1, \ldots, x_n) \in]-1, 1[^n \right\},$$

où les $\mathbf{b}_i$ forment une base quelconque du réseau. Le théorème s'étend au cas d'égalité $\mu(C) \geq 2^n \mathrm{vol}(L)$, lorsque C est supposé de plus compact. En l'appliquant à des boules fermées, on obtient ainsi :

Corollaire 3.1 *Tout réseau L de $\mathbb{R}^n$ de dimension d contient un élément non nul $\mathbf{x}$ tel que*

$$\|\mathbf{x}\| \leq 2 \left(\frac{\det(L)}{v_d} \right)^{\frac{1}{d}},$$

v_d désignant le volume de la boule unité fermée de $\mathbb{R}^d$.

On a un résultat analogue pour la norme infinie, bien que la norme infinie ne soit pas un invariant géométrique :

Théorème 3.5 *Soit L un réseau de dimension d. Alors il existe un point non nul $\mathbf{x}$ de L tel que :*

$$\|\mathbf{x}\|_\infty \leq \det(L)^{1/d}.$$

Remarquons que cette borne est atteinte dans le cas $L = \mathbb{Z}^d$.

Minima successifs

Le théorème de Minkowski nous amène naturellement à nous intéresser à la plus petite boule contenant un vecteur non nul d'un réseau, ou plus généralement, à la plus petite boule contenant un nombre donné de vecteurs linéairement indépendants d'un réseau. On définit ainsi les *minima successifs* d'un réseau d-dimensionnel L de $\mathbb{R}^n$ comme étant les nombres réels positifs $\lambda_1(L), \ldots, \lambda_d(L)$ où $\lambda_r(L)$ est la borne inférieure des nombres réels λ tels qu'il existe r vecteurs linéairement indépendants de L de norme inférieure ou égale à λ. Cette borne inférieure est atteinte, puisqu'un réseau est discret. Et clairement : $\lambda_1(L) \leq \lambda_2(L) \leq \cdots \leq \lambda_d(L)$.

Le premier minimum $\lambda_1(L)$ est aussi appelé *norme* du réseau L : c'est la norme du plus court vecteur non nul de L, que l'on notera $\|L\|$ lorsqu'on ne s'intéressera pas aux autres minima. C'est aussi la distance minimale entre deux points distincts de L. L'*invariant d'Hermite*[4] de L est par définition :

$$\gamma(L) = \left(\frac{\lambda_1(L)}{\det(L)^{1/\dim L}} \right)^2 .$$

Une conséquence du théorème de Minkowski est qu'à dimension n fixée, l'invariant $\gamma(L)$ est borné lorsque L parcourt tous les réseaux de dimension n. On peut ainsi définir la *constante d'Hermite de rang* n, notée γ_n, comme étant la borne supérieure de $\gamma(L)$ pour L parcourant l'ensemble des réseaux de dimension n.[5] Cette constante est atteinte, et les réseaux totaux qui l'atteignent font l'objet de nombreuses recherches (voir par exemple [23], ce sont les réseaux dits critiques). On sait en outre que γ_n^n est un rationnel. Les valeurs exactes de γ_n ne sont connues que pour $n \leq 8$. Le théorème de Minkowski permet de montrer :

$$\forall n,\ \gamma_n \leq 1 + \frac{n}{4}.$$

Le meilleur encadrement asymptotique connu de la constante d'Hermite est le suivant :

$$\frac{n}{2\pi e} + \frac{\log(\pi n)}{2\pi e} + o(1) \leq \gamma_n \leq \frac{1.744n}{2\pi e}(1 + o(1)).$$

La détermination exacte de la constante d'Hermite est l'un des problèmes fondamentaux de la géométrie des nombres. Il est relié au problème classique des empilements de sphères : quelle est la densité maximale possible pour une union de boules de rayon fixé qui ne se recouvrent pas (*i.e.*, ont au plus un point commun) dans $\mathbb{R}^n$? Minkowski a montré plus généralement :

Théorème 3.6 (Minkowski) *Pour tout réseau L de dimension d, les minima successifs vérifient pour tout $r \leq d$:*

$$\prod_{i=1}^{r} \lambda_i(L) \leq \sqrt{\gamma_d^r} \det(L)^{r/d}.$$

On peut facilement montrer par récurrence qu'il existe une famille libre du réseau qui réalise les minima successifs : il existe des vecteurs linéairement indépendants $\mathbf{a}_1, \ldots, \mathbf{a}_d$

[4] L'invariant d'Hermite est élevé au carré pour des raisons historiques.

[5] Hermite fut le premier à démontrer l'existence d'une telle constante, dans le langage des formes quadratiques. Les techniques d'Hermite fournissent une majoration exponentielle en n, tandis que celles de Minkowski donnent une majoration linéaire en n.

de L telle que pour tout i, $\|\mathbf{a}_i\| = \lambda_i$. Mais curieusement, une telle famille libre n'est pas nécessairement une base du réseau dès que la dimension est supérieure ou égale à 4. En effet, considérons le réseau L formé par tous les vecteurs de $\mathbb{Z}^4$ tels que la somme des coordonnées soit paire. On peut facilement voir que les minima de L sont tous égaux à $\sqrt{2}$, et on peut noter au passage que $\mathrm{vol}(L) = 2$. Pourtant, les vecteurs lignes de la matrice suivante sont linéairement indépendants, atteignent les minima de L, mais ne constituent pas une base de L :

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

En outre, il n'existe pas en général de base réalisant les minima successifs d'un réseau, dès que la dimension est supérieure ou égale à 5. Ce résultat non intuitif fut formulé pour la première fois par Korkine et Zolotareff dans le langage des formes quadratiques. Considérons le réseau L engendré par les vecteurs $\mathbf{e}_1, \ldots, \mathbf{e}_n$ de la base canonique de $\mathbb{R}^n$, et le vecteur $\mathbf{h}$ dont toutes les coordonnées sont égales à $\frac{1}{2}$. On remarque que $\{\mathbf{h}, \mathbf{e}_2, \ldots, \mathbf{e}_n\}$ est une base de L, puisque $\mathbf{e}_1 = 2\mathbf{h} - \sum_{i=2}^{n}$. Les $\mathbf{e}_i$ ne forment cependant pas une base de L, puisqu'ils engendrent $\mathbb{Z}^n$, qui ne contient pas $\mathbf{h}$. Or $\|\mathbf{h}\| = \sqrt{n/4}$. Donc, si $n > 4$, tous les minima successifs de L sont égaux à 1, et ils sont atteints par les $\mathbf{e}_i$, mais ces n vecteurs linéairement indépendants ne forment pas une base.

D'un point de vue mathématique, on connaît très peu de choses sur la distribution des minima successifs, si ce n'est en dimension 2. De fait, on se contente dans la pratique d'heuristiques « intuitives ». Moralement, lorsqu'un réseau L de dimension d est régulier, ses minima successifs sont du même ordre de grandeur, celui de $\sqrt{\gamma_d}\det(L)^{1/d}$, qui vaut à un facteur multiplicatif près $\sqrt{d}\det(L)^{1/d}$.

L'algorithme LLL

En dimension supérieure à 2, tout réseau admet une infinité de bases. Mais certaines bases sont plus intéressantes que d'autres, par exemples celles qui sont formées de vecteurs relativement courts et quasi orthogonaux : on dit qu'elles sont *réduites*. On a vu que des phénomènes pour le moins curieux se produisaient dès que la dimension du réseau devenait supérieure à 4. En particulier, il n'y a pas nécessairement de base atteignant les minima successifs, ce qui rend impossible la définition d'une notion de réduction « optimale » . Cette situation a donné naissance à plusieurs notions de *base réduite*, selon les propriétés recherchées.

D'un point de vue algorithmique, une notion de réduction n'est pertinente que si l'on sait trouver de telles bases réduites en un temps raisonnable. En ce sens, la première notion de réduction intéressante fut proposée il y a vingt ans par Lenstra, Lenstra et Lovász [22]. Cette réduction est aujourd'hui appelée réduction LLL. Nous ne définirons pas formellement ce qu'est la réduction LLL (voir en annexe, ou bien [25, Chapitre 2] et [15, 9]), et nous nous contenterons de citer le principal résultat algorithmique obtenu par [22] :

Théorème 3.7 (LLL) *Il existe un algorithme qui, étant donné en entrée une base $(\mathbf{c}_1, \ldots, \mathbf{c}_r)$ d'un réseau L de $\mathbb{Z}^n$, fournit en temps polynomial en n, r et la taille des $\mathbf{c}_i$, une base $(\mathbf{b}_1, \ldots, \mathbf{b}_r)$ de L vérifiant :*

1. $\|\mathbf{b}_1\| \leq 2^{(r-1)/4}(\mathrm{vol}L)^{1/r}$.
2. *Pour tout i :* $\|\mathbf{b}_i\| \leq 2^{(r-1)/2}\lambda_i(L)$.

3. $\|\mathbf{b}_1\| \cdot \cdots \cdot \|\mathbf{b}_r\| \leq 2^{\binom{r}{2}/2} \mathrm{vol} L$.

La constante 2 peut en fait être remplacée par n'importe quelle constante $> 4/3$. L'algorithme obtenu est connu sous le nom d'algorithme LLL. Il permet donc de trouver un vecteur (non nul) relativement court de n'importe quel réseau : c'est en quelque sorte une version constructive du théorème de Minkowski sur le premier minimum d'un réseau.

3.6.2 Le théorème de Coppersmith

On ne sait pas extraire des racines e-ièmes modulo n sans factoriser n. Plus généralement, on ne sait pas résoudre une équation polynomiale modulo n sans factoriser n (et ensuite utiliser les restes chinois). En 1996, Coppersmith [10] a démontré, à l'aide de l'algorithme LLL, que l'on pouvait trouver efficacement toutes les petites solutions d'équation polynomiale modulo n sans factoriser n :

Théorème 3.8 (Coppersmith) *Soient $P(x)$ un polynôme unitaire de degré δ à coefficients entiers, et n un entier de factorisation inconnue. Alors on peut trouver en temps polynomial en $(\log n, \delta)$ tous les entiers x_0 tels que $P(x_0) \equiv 0 \ (mod\, n)$ et $|x_0| \leq n^{1/\delta}$.*

On va montrer ce théorème en adoptant la présentation de [19]. On trouvera une discussion plus approfondie du théorème de Coppersmith dans [26, Section 6].

L'idée de Coppersmith est de réduire le problème de la recherche des petites racines modulaires au problème (facile) de la résolution d'équations polynomiales sur $\mathbb{Z}$. Plus précisément, Coppersmith utilise la réduction de réseau pour trouver une équation polynomiale (sur $\mathbb{Z}$) satisfaite par toutes les petites racines modulaires de P. Intuitivement, on essaie de linéariser toutes les équations de la forme $x^i P(x)^j \equiv 0 \pmod{n^j}$ pour des valeurs entières appropriées de i et j. De telles équations sont satisfaites par n'importe quelle solution de $P(x) \equiv 0 \pmod n$. Les petites solutions x_0 donnent naissance à des solutions inhabituellement petites au système linéaire obtenu. Pour transformer des équations modulaires en des équations entières, on utilise le lemme élémentaire suivant, avec la notation $\|r(x)\| = \sqrt{\sum a_i^2}$ pour tout polynôme $r(x) = \sum a_i x^i \in \mathbb{Q}[x]$:

Lemme 3.2 *Soit $r(x) \in \mathbb{Q}[x]$ un polynôme de degré $< m$ et soit X un entier positif. Supposons $\|r(xX)\| < 1/\sqrt{m}$. Si $r(x_0) \in \mathbb{Z}$ avec $|x_0| < X$, alors $r(x_0) = 0$ sur $\mathbb{Z}$.*

Ce lemme provient du fait que tout entier suffisamment petit est nécessairement nul. L'astuce consiste à présent à se fixer un paramètre h et à considérer les $m = (h+1)\delta$ polynômes $q_{u,v}(x) = x^u (P(x)/n)^v$, où $0 \leq u \leq \delta - 1$ et $0 \leq v \leq h$. Remarquons que l'évaluation de $q_{u,v}(x)$ en n'importe quelle racine x_0 de $P(x)$ modulo n est nécessairement un entier. Et cela reste vrai si l'on prend toute combinaison linéaire à coefficients entiers des $q_{u,v}(x)$. Si une telle combinaison $r(x)$ satisfait en outre $\|r(xX)\| < 1/\sqrt{m}$, alors le lemme 3.2 nous assure que la résolution de l'équation $r(x) = 0$ sur $\mathbb{Z}$ fournira toutes les racines de $P(x)$ modulo n inférieures à X en valeur absolue. Cela suggère la recherche d'un vecteur court dans le réseau correspondant aux $q_{u,v}(xX)$. Plus précisément, définissons la matrice $(m \times m)$ M dont la i-ième ligne est constituée des coefficients de $q_{u,v}(xX)$, en commençant par les termes de plus bas degré, et où $v = \lfloor (i-1)/\delta \rfloor$ et $u = (i-1) - \delta v$. On remarque que M est triangulaire inférieure, et un calcul élémentaire montre que $\mathrm{vol}(M) = X^{m(m-1)/2} n^{-mh/2}$. Appliquons une réduction LLL au réseau total engendré par les lignes de M. Le premier vecteur de la base réduite obtenue correspond à un polynôme de la forme $r(xX)$, et a pour norme euclidienne $\|r(xX)\|$. On sait d'après le théorème 3.7 que :

$$\|r(xX)\| \leq 2^{(m-1)/4} \mathrm{vol}(M)^{1/m} = 2^{(m-1)/4} X^{(m-1)/2} n^{-h/2}.$$

Et rappelons qu'il faut $\|r(xX)\| \leq 1/\sqrt{n}$ pour appliquer le lemme 3.2. Par conséquent, pour tout h donné, la méthode garantit de trouver toutes les racines modulaires inférieures à X si :

$$X \leq \frac{1}{\sqrt{2}} n^{h/(m-1)} m^{-1/(m-1)}.$$

La limite de cette borne supérieure, lorsque h tend vers l'infini est $\frac{1}{\sqrt{2}} n^{1/\delta}$. Le théorème 3.8 s'en déduit par un choix approprié de h.

En généralisant légèrement le lemme 3.2, et après quelques calculs asymptotiques, on peut obtenir une généralisation ion du théorème 3.8 :

Théorème 3.9 *Soit $\alpha = a/b$ un rationnel tel que $0 \leq \alpha \leq 1$. Soient $P(x)$ un polynôme unitaire de degré δ à coefficients entiers, et n un entier de factorisation inconnue. Alors on peut trouver en temps polynomial en $(\log n, \log a, \log b, \delta)$ tous les entiers x_0 tels que $|x_0| \leq n^{\alpha^2/\delta}$ et que le pgcd de $P(x_0)$ avec n soit $\geq n^\alpha$.*

Le théorème 3.8 n'est alors que le cas particulier $\alpha = 1$.

3.6.3 Application au RSA à base d'identité

Considérons à nouveau un module RSA $n = pq$, et des exposants public et privé e et d : $ed \equiv 1 \pmod{\varphi(n)}$.

On a vu que l'utilisation des restes chinois pouvait accélérer le déchiffrement, mais cette utilisation nécessite le stockage des nombres premiers p et q. Pour diminuer ce stockage, il fut proposé de coder dans p et q l'identité de l'utilisateur, en rajoutant suffisamment d'aléa. Par exemple, supposons que la moitié des bits de poids forts de p soit connue : $p = p_0 + \varepsilon$ où p_0 est connu, et $0 \leq \varepsilon \leq n^{1/4}$ inconnu. Supposons également que p et q sont équilibrés, par exemple que $p \geq n^{1/2}$.

Considérons alors le polynôme $P(x) = p_0 + x$. Le pgcd de $P(\varepsilon)$ avec n est supérieur à $n^{1/2}$, avec $0 \leq \varepsilon \leq n^{1/4}$. En appliquant le théorème 3.9 avec $\alpha = 1/2$, on peut donc retrouver p et q en temps polynomial à partir uniquement de p_0 et n.

3.6.4 Application au RSA à petit exposant public

Chiffrement stéréotypé

Supposons que l'on chiffre des messages stéréotypés, c'est-à-dire qu'une grande partie du message soit déjà connue. Par exemple, supposons que le message m soit de la forme « Le mot de passe d'aujourd'hui est ? ? ? », où « ? ? ? » désigne quelque chose d'inconnu. On peut donc écrire le message m sous la forme $m = m_0 + \varepsilon$ où m_0 est connu et ε inconnu mais petit. On connaît aussi le chiffré RSA $c = m^e \bmod n$. En utilisant la formule du binôme, on obtient une équation polynomiale unitaire en ε, modulo n. Donc, d'après le théorème de Coppersmith, on peut retrouver ε (et donc m) en temps polynomial en e et $\log n$, à partir uniquement de c, n et e. Plus généralement, on a en fait démontré le résultat suivant :

Théorème 3.10 *Soient $\mathsf{pk} = (n, e)$ et $\mathsf{sk} = d$ des clefs RSA, avec $n = pq$. Soit $m \in \mathbb{Z}_n$ un message chiffré en $c = m^e \bmod n$. Si l'on connaît entièrement m à l'exception d'au plus $\log_2(n)/e$ bits consécutifs, alors on peut retrouver l'intégralité de m en temps polynomial en $(e, \log n)$ à partir de c, n et e.*

L'attaque sur les messages courts (section 3.3.3) n'est autre que le cas particulier où m est petit. En d'autres termes, le théorème 3.8 est trivial lorsque $P(x)$ est de la forme $P(x) = x^\delta + c$.

Chiffrement multiple

On reprend le scénario du broadcast (section 3.3.4). Supposons donc qu'Alice communique avec r personnes disposant chacune d'une clef publique RSA $\mathsf{pk}_i = (n_i, e)$ distincte, mais utilisant le même petit exposant public e. On a vu qu'Alice ne devait pas chiffrer directement le même message m à ces r personnes en utilisant la clef respective de ces r personnes. Pour empêcher l'attaque élémentaire vue en section 3.3.4, Alice peut par exemple rajouter de l'information spécifique à chaque destinataire, mais publiquement connue. Par exemple, on peut imaginer qu'Alice rajoute en tête de chaque message l'identité du destinataire. Plus généralement, on peut supposer qu'Alice transforme le message m qu'elle souhaite envoyer en $\alpha_i m + \beta_i$ où α_i et β_i sont des valeurs publiques dépendant de l'utilisateur. Ainsi, Alice transmet les chiffrés $c_i = (\alpha_i m + \beta_i)^e \bmod n_i$.

Si les modules n_i ne sont pas premiers entre eux, alors on peut casser certaines des clefs RSA par simple pgcd. Sinon, on peut par restes chinois calculer une équation polynomiale en m de degré e, modulo $\prod_i n_i$. Cette équation polynomiale peut être rendue unitaire, ou alors l'un des n_i peut être factorisé. On en déduit que l'on peut retrouver l'intégralité de m si le nombre de destinataires est $\geq e$.

Chiffrement avec petit aléa

On a vu en section 3.3 qu'il fallait nécessairement rajouter de l'aléa avant de chiffrer des messages. Supposons donc que l'on se contente de rajouter quelques bits d'aléa au message. Par exemple, supposons que le message après formatage soit de la forme $2^k m + r$ où m est le message original, k est connu, et r est inconnu et représente l'aléa choisi à chaque chiffrement. Il semble réaliste de supposer en outre qu'un attaquant puisse disposer de plusieurs chiffrés d'un même message avec des aléas différents. Supposons ainsi qu'un attaquant connaisse $c_1 = (2^k m + r_1)^e \bmod n$ et $c_2 = (2^k m + r_2)^e \bmod n$. Posons $r_3 = r_2 - r_1$ qui est petit. On peut alors écrire $c_1 = M^e \bmod n$ et $c_2 = (M + r_3)^e \bmod n$, en posant $M = 2^k m + r_1$. On obtient donc deux équations polynomiales satisfaites par (M, r_3) dont M est une racine commune. En calculant le résultant de ces deux équations par rapport à la variable M, on en déduit une équation polynomiale unitaire de degré e^2 en r_3. Ainsi, le théorème de Coppersmith nous assure que si $|r_3| \leq n^{1/e^2}$ (ce qui est notamment vrai si $0 \leq r_1 \leq n^{1/e^2}$ et $0 \leq r_2 \leq n^{1/e^2}$), alors on peut calculer r_3 en temps polynomial. En appliquant ensuite l'attaque de la section 3.5.1, on en déduit le message original m.

3.6.5 Application au RSA à petit exposant privé

On suppose cette fois d petit. L'attaque de Wiener fonctionne lorsque $d = O(n^{1/4})$, $e \leq n$, p et q équilibrés. Rappelons que : $ed \equiv 1 \pmod{\varphi n}$. Il existe donc un entier k tel que :

$$ed = 1 + k\varphi n.$$

De plus, $\varphi n = (p-1)(q-1) = n - (p+q) + 1 = n - s + 1$ en posant $s = p + q$. Ainsi :

$$ed = 1 + k(n - s + 1),$$

d'où :

$$1 + k(n - s + 1) \equiv 0 \pmod{e}.$$

Les deux inconnues de cette équation polynomiale modulaire sont relativement petites : $s \approx n^{1/2}$ (p et q étant supposés petits) et k est de l'ordre de d.

Malheureusement, on ne peut appliquer directement le théorème de Coppersmith puisqu'il n'est démontré que pour des polynômes à une seule variable. Toutefois, ce théorème peut heuristiquement s'étendre à des polynômes multivariables. En effet, au lieu de rechercher un seul vecteur court dans le réseau construit dans la preuve du théorème de Coppersmith, on peut en fait en trouver plusieurs en un temps raisonnable. On obtient ainsi plusieurs équations polynomiales sur $\mathbb{Z}$ satisfaites par toutes les petites racines modulaires. En calculant des résultants appropriés de ces polynômes, on peut réduire le nombre d'inconnues jusqu'à se ramener au cas d'une seule équation polynomiale sur $\mathbb{Z}$ à une seule variable. Malheureusement, rien ne prouve que les résultants ne soient pas nuls : les équations obtenues à partir du réseau sont linéairement indépendantes en tant que vecteurs, mais ne sont pas nécessairement algébriquement indépendantes. Par conséquent, la méthode n'est qu'heuristique. Trouver des conditions suffisantes généralisant le théorème de Coppersmith aux polynômes à plusieurs variables est l'un des problèmes ouverts majeurs en cryptanalyse.

Cependant, Boneh et Durfee [4] ont appliqué cette méthode au cas de l'équation polynomiale à deux variables du problème du RSA à petit exposant privé d. Ils ont obtenu à l'aide de réseaux non totaux une amélioration heuristique de l'attaque de Wiener : si p et q sont équilibrés, e est de l'ordre de n, et $d \leq n^{0.292}$, alors on peut heuristiquement factoriser n. En dépit de son caractère heuristique, cette attaque fonctionne très bien en pratique.

Bibliographie

[1] Bleichenbacher D.. – Chosen Ciphertext Attacks against Protocols Based on RSA Encryption Standard PKCS #1. In *Proc. of Crypto '98. Lecture Notes in Computer Science*, volume 1462, pp. 1–12. – Springer-Verlag, 1998.

[2] Boneh D. – Twenty years of attacks on the RSA cryptosystem. *Notices of the AMS*, vol. 46, N° 2, 1999, pp. 203–213.

[3] Boneh D., DeMillo R. et Lipton R. – On the importance of checking cryptographic protocols for faults. In *Proc. of Eurocrypt '97. Lecture Notes in Computer Science*, volume 1233, pp. 37–51. – Springer-Verlag, 1997.

[4] Boneh D. et Durfee G. – Cryptanalysis of RSA with private key d less than $N^{0.292}$. In *Proc. of Eurocrypt '99. Lecture Notes in Computer Science*, volume 1592, pp. 1–11. – Springer-Verlag, 1999.

[5] Boneh D., Durfee G. et Howgrave-Graham N. – Factoring $N = p^r q$ for large r. In *Proc. of Crypto '99. Lecture Notes in Computer Science*, volume 1666. – Springer-Verlag, 1999.

[6] Boneh D., Joux A. et Nguyen P. Q. – Why textbook ElGamal and RSA encryption are insecure. In *Proc. of Asiacrypt '00. LNCS*, volume 1976. – Springer-Verlag, 2000.

[7] Boneh D. et Venkatesan R. – Breaking RSA may not be equivalent to factoring. In *Proc. of Eurocrypt '98. Lecture Notes in Computer Science*, volume 1233, pp. 59–71. – Springer-Verlag, 1998.

[8] CESG. – The history of non-secret encryption. – Rapports confidentiels du *Communinications-Electronics Security Group* sous l'autorité du gouvernement britannique, datant des années 70, et mis à disposition à l'adresse `http://www.cesg.gov.uk/about/nsecret/home.htm`.

[9] Cohen H. – *A Course in Computational Algebraic Number Theory*. – Springer-Verlag, 1995. Deuxième édition.

[10] Coppersmith D. – Small solutions to polynomial equations, and low exponent RSA vulnerabilities. *J. of Cryptology*, vol. 10, nN° 4, 1997, pp. 233–260.

[11] Coppersmith D., Franklin M., Patarin J. et Reiter M. – Low-exponent RSA with related messages. In *Proc. of Eurocrypt '96. Lecture Notes in Computer Science*, volume 1070. – Springer-Verlag, 1996.

[12] Coupé C., Nguyen P. et Stern J. – The effectiveness of lattice attacks against low-exponent RSA. In *Proc. of PKC'98. Lecture Notes in Computer Science*, volume 1431. – Springer-Verlag, 1999.

[13] Diffie W. et Hellman M. E. – New directions in cryptography. *IEEE Trans. Inform. Theory*, vol. IT-22, Nov 1976, pp. 644–654.

[14] Durfee G. et Nguyen P. Q. – Cryptanalysis of the RSA schemes with short secret exponent from Asiacrypt '99. In *Proc. of Asiacrypt '00*. IACR, *LNCS*, volume 1976. – Springer-Verlag, 2000.

[15] Grötschel M., Lovász L. et Schrijver A. – *Geometric Algorithms and Combinatorial Optimization*. – Springer-Verlag, 1993.

[16] Gruber M. et Lekkerkerker C. G. – *Geometry of Numbers*. – North-Holland, 1987.

[17] Hardy G. H. et Wright E. M. – *An introduction to the theory of numbers*. – Oxford university press, 1979.

[18] Hastad J. – Solving simultaneous modular equations of low degree. *SIAM J. Comput.*, vol. 17, nN° 2, April 1988, pp. 336–341.

[19] Howgrave-Graham N. – Finding small roots of univariate modular equations revisited. In *Cryptography and Coding. Lecture Notes in Computer Science*, volume 1355, pp. 131–142. – Springer-Verlag, 1997.

[20] Kocher P.. – Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *Proc. of Crypto '96. Lecture Notes in Computer Science*, volume 1109, pp. 104–113. – Springer-Verlag, 1996.

[21] Lenstra A. K. et Lenstra, Jr. H. W. – *The Development of the Number Field Sieve*. – Springer-Verlag, 1993, *Lecture Notes in Mathematics*, vol. 1554.

[22] Lenstra A. K., Lenstra, Jr. H. W. et Lovász L. – Factoring polynomials with rational coefficients. *Mathematische Ann.*, vol. 261, 1982, pp. 513–534.

[23] Martinet J. – *Les Réseaux Parfaits des Espaces Euclidiens*. – Editions Masson, 1996. Traduction en cours chez Springer-Verlag.

[24] Menezes A., Oorschot P. V. et Vanstone S. – *Handbook of Applied Cryptography*. – CRC Press, 1997.

[25] Nguyen P. Q. – *La Géométrie des Nombres en Cryptologie*. – Thèse de Doctorat, Université Paris 7, 1999.

[26] Nguyen P. Q. et Stern J. – The two faces of lattices in cryptology. In *Proc. of CALC '01. Lecture Notes in Computer Science*, volume 2146. – Springer-Verlag, 2001.

[27] Rivest R. L., Shamir A. et Adleman L. M. – A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, vol. 21, nN° 2, 1978, pp. 120–126.

[28] RSA Security– Voir *www.rsa.com*.

[29] Siegel C. L. – *Lectures on the Geometry of Numbers*. – Springer-Verlag, 1989.

[30] Wiener M. – Cryptanalysis of short RSA secret exponents. *IEEE Trans. Inform. Theory*, 36(3) :553–558, 1990.

Annexe

3. A L'algorithme LLL

3. A.1 L'orthogonalisation de Gram-Schmidt

Soit $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ une famille libre de $\mathbb{R}^n$. On rappelle que le procédé d'orthogonalisation de Gram-Schmidt construit par récurrence une famille orthogonale $(\mathbf{b}_1^\star, \ldots, \mathbf{b}_d^\star)$, en prenant $\mathbf{b}_i^\star = \pi_i(\mathbf{b}_i)$, où π_i est la projection orthogonale sur le supplémentaire orthogonal du sous-espace $\sum_{j=1}^{i-1} \mathbb{R}\mathbf{b}_j = \sum_{j=1}^{i-1} \mathbb{R}\mathbf{b}_j^\star$. Cela revient à la formulation suivante :

$$\mathbf{b}_i^\star = \mathbf{b}_i - \sum_{j=1}^{i-1} \mu_{i,j}\mathbf{b}_j^\star, \text{ avec } \mu_{i,j} = \frac{\langle \mathbf{b}_i, \mathbf{b}_j^\star \rangle}{\|\mathbf{b}_j^\star\|^2}.$$

Typiquement, les derniers vecteurs $\mathbf{b}_i^\star$ sont les plus courts. L'orthogonalisation peut s'effectuer en temps polynomial (lorsque les $\mathbf{b}_i$ sont rationnels). La matrice dont les colonnes expriment les $\mathbf{b}_i^\star$ selon les $\mathbf{b}_i$ est donc triangulaire supérieure, et sa diagonale n'est composée que de 1. Ainsi, le déterminant de Gram vaut :

$$\Delta(\mathbf{b}_1, \ldots, \mathbf{b}_d) = \prod_{i=1}^{d} \|\mathbf{b}_i^\star\|^2. \tag{3.1}$$

On a en outre :

$$\|\mathbf{b}_i^\star\|^2 \leq \|\mathbf{b}_i\|^2 = \|\mathbf{b}_i^\star\|^2 + \sum_{j=1}^{i-1} |\mu_{i,j}|^2 \|\mathbf{b}_j^\star\|^2. \tag{3.2}$$

Cela signifie que les $\mathbf{b}_i^\star$ encadrent en quelque sorte les $\mathbf{b}_i$. De plus, la famille de Gram-Schmidt d'une base d'un réseau permet de minorer les minima successifs :

Lemme 3.3 *Si $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ est une base d'un réseau L, alors pour tout $1 \leq i \leq d$:*

$$\lambda_i(L) \geq \min_{i \leq j \leq d} \|\mathbf{b}_j^\star\|.$$

Preuve. On sait qu'il existe une famille libre $(\mathbf{a}_1, \ldots, \mathbf{a}_d)$ de L réalisant les minima successifs : pour tout i, $\|\mathbf{a}_i\| = \lambda_i$. Décomposons ces vecteurs selon les $\mathbf{b}_j$:

$$\mathbf{a}_i = \sum_{j=1}^{d} a_{i,j}\mathbf{b}_j, a_{i,j} \in \mathbb{Z}.$$

On pose $a_i = \max\{j \ : a_{i,j} \neq 0\}$. Soit à présent i dans $\{1, \ldots, d\}$. Supposons par l'absurde que $a_k < i$ pour tout $k \leq i$. Alors les vecteurs linéairement indépendants $\mathbf{a}_1, \ldots, \mathbf{a}_i$ appartiennent au sous-espace engendré par $\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}$, qui est de dimension $i-1$: contradiction. Donc il existe $k \leq i$ tel que $a_k \geq i$. On en déduit $\lambda_i \geq \lambda_k = \|\mathbf{a}_k\|$ avec :

$$\mathbf{a}_k = \sum_{j=1}^{a_k} a_{k,j}\mathbf{b}_j = \sum_{j=1}^{a_k} a_{k,j}^\star \mathbf{b}_j^\star, \text{ où } a_{k,j}^\star \in \mathbb{R}.$$

Par Pythagore : $\|\mathbf{a}_k\| \geq |a^\star_{k,j}| \times \|\mathbf{b}^\star_j\|$. Or on a en particulier $a^\star_{k,a_k} = a_{k,a_k} \in \mathbb{Z}^*$, d'où :

$$\lambda_i \geq \lambda_k = \|\mathbf{a}_k\| \geq \|\mathbf{b}^\star_{a_k}\|, \text{avec } a_k \geq i.$$

□

Pour approcher le premier minimum d'un réseau, il suffit donc de déterminer une base telle que les $\mathbf{b}^\star_i$ ne soient pas trop éloignés les uns des autres. Ainsi, aucun des $\mathbf{b}^*_i$ ne pourra être très éloigné de $\mathbf{b}^*_1 = \mathbf{b}_1$, et donc $\|\mathbf{b}_1\|$ sera relativement proche de $\lambda_1(L)$. Dans ce cas, on a même chacun des $\|\mathbf{b}_i\|$ qui n'est pas trop éloigné de $\lambda_i(L)$, si les $|\mu_{i,j}|$ sont petits, d'après (3.2). On va à présent introduire la notion de réduction faible, qui permet justement de majorer les $|\mu_{i,j}|$.

3. A.2 La réduction faible

On dit qu'une base $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ d'un réseau est *faiblement réduite* si son orthogonalisation de Gram-Schmidt vérifie : pour tout $1 \leq j < i \leq d$,

$$|\mu_{i,j}| \leq \frac{1}{2}.$$

En d'autres termes, la base est faiblement orthogonale. On a alors d'après (3.2) :

$$\|\mathbf{b}^\star_i\|^2 \leq \|\mathbf{b}_i\|^2 \leq \|\mathbf{b}^\star_i\|^2 + \frac{1}{4}\sum_{j=1}^{i-1} \|\mathbf{b}^\star_j\|^2. \tag{3.3}$$

La figure 3.3 fournit un algorithme très simple permettant de réduire faiblement une base, tout en conservant la famille de Gram-Schmidt. Soient $j < i$. On remarque qu'une opé-

Entrée : une base $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ d'un réseau L.
Sortie : la base $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ est faiblement réduite.

1. Calculer les coefficients de Gram-Schmidt $\mu_{i,j}$.
2. Pour i allant de 2 à n
3. Pour j allant de $i-1$ à 1
4. $\mathbf{b}_i \longleftarrow \mathbf{b}_i - \lceil \mu_{i,j} \rfloor \mathbf{b}_j$
5. Pour k allant de 1 à j
6. $\mu_{i,k} \longleftarrow \mu_{i,k} - \lceil \mu_{i,j} \rfloor \mu_{j,k}$

FIG. 3.3 – Un algorithme de réduction faible.

ration $\mathbf{b}_i \longleftarrow \mathbf{b}_i - \lceil \mu_{i,j} \rfloor \mathbf{b}_j$ ne modifie aucun des vecteurs de Gram-Schmidt. Donc seuls les coefficients de la forme $\mu_{i,k}$ (avec $k \leq j$) peuvent éventuellement être modifiés. Et ces nouveaux coefficients valent :

$$\mu'_{i,k} = \frac{\langle \mathbf{b}_i - \lceil \mu_{i,j} \rfloor \mathbf{b}_j, \mathbf{b}^\star_k \rangle}{\|\mathbf{b}^\star_k\|^2} = \mu_{i,k} - \lceil \mu_{i,j} \rfloor \frac{\langle \mathbf{b}_j, \mathbf{b}^\star_k \rangle}{\|\mathbf{b}^\star_k\|^2} = \begin{cases} \mu_{i,j} - \lceil \mu_{i,j} \rfloor & \text{si } j = k\,; \\ \mu_{i,k} - \lceil \mu_{i,j} \rfloor \mu_{j,v} & \text{si } j > k\,; \\ \mu_{i,k} & \text{si } j < k. \end{cases}$$

Ainsi, on est assuré que le nouveau $\mu_{i,j}$ sera inférieur à $\frac{1}{2}$ en valeur absolue. En outre, seuls les coefficients $\mu_{i,j}$ avec $k \leq j$ peuvent être modifiés. Par conséquent, l'algorithme de réduction faible est correct, et il possède deux propriétés intéressantes :

– Les vecteurs $\mathbf{b}_1^\star, \ldots, \mathbf{b}_d^\star$ restent inchangés.
– Les valeurs des coefficients $\mu_{i,j}$ sont automatiquement mises à jour au cours de l'algorithme.

3. A.3 L'algorithme de Lenstra-Lenstra-Lovász

Soit un réel δ dans $]\frac{1}{4}, 1]$. Une base ordonnée $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ de L est dite LLL-*réduite* à un facteur δ si elle est faiblement réduite, et si elle satisfait la *condition de Lovász* : pour tout $1 < i \leq d$,

$$\left\|\mathbf{b}_{i+1}^\star + \mu_{i+1,i}\mathbf{b}_i^\star\right\|^2 \geq \delta \|\mathbf{b}_i^\star\|^2.$$

Expliquons cette mystérieuse condition. Comme l'orthogonalisation de Gram-Schmidt dépend de l'ordre des éléments, ses vecteurs changent si $\mathbf{b}_i$ et $\mathbf{b}_{i+1}$ sont interchangés ; en fait, seuls $\mathbf{b}_i^\star$ et $\mathbf{b}_{i+1}^\star$ peuvent éventuellement changer. Et le nouveau $\mathbf{b}_i^\star$ n'est autre que $\mathbf{b}_{i+1}^\star + \mu_{i+1,i}\mathbf{b}_i^\star$, donc la condition de Lovász signifie qu'en interchangeant $\mathbf{b}_i$ et $\mathbf{b}_{i+1}$, la longueur de $\mathbf{b}_i^\star$ ne diminue pas trop, la perte étant quantifiée par δ : on ne peut pas gagner beaucoup sur $\|\mathbf{b}_i^\star\|$ en les échangeant. La valeur la plus naturelle est donc $\delta = 1$ mais on ne connaît alors pas d'algorithme prouvé polynomial pour obtenir une base LLL-réduite. La réduction LLL fut initialement présentée avec le facteur $\delta = \frac{3}{4}$, si bien que par réduction LLL dans la littérature, on entend souvent réduction LLL avec ce facteur-là.

La condition de Lovász peut aussi s'écrire de façon équivalente : pour tout i,

$$\|\mathbf{b}_{i+1}^\star\|^2 \geq \left(\delta - \mu_{i+1,i}^2\right) \|\mathbf{b}_i^\star\|^2.$$

La réduction LLL garantit ainsi que chaque $\mathbf{b}_{i+1}^\star$ ne peut être beaucoup plus court que $\mathbf{b}_i^\star$: la décroissance est au pire géométrique. Le résultat suivant en découle :

Théorème 3.11 *On suppose $\frac{1}{4} < \delta \leq 1$, et on note $\alpha = 1/(\delta - \frac{1}{4})$. Soit $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ une base LLL-réduite à un facteur δ d'un réseau de L de $\mathbb{R}^n$. Alors :*

1. $\|\mathbf{b}_1\| \leq \alpha^{(d-1)/4}(\mathrm{vol}L)^{1/d}$.
2. *Pour tout i :* $\|\mathbf{b}_i\| \leq \alpha^{(d-1)/2}\lambda_i(L)$.
3. $\|\mathbf{b}_1\| \times \cdots \times \|\mathbf{b}_d\| \leq \alpha^{\binom{d}{2}/2}\mathrm{vol}L$.

Preuve. La base étant faiblement réduite, la condition de Lovász implique que pour tout j, $\|\mathbf{b}_{j+1}^\star\|^2 \geq (\delta - \frac{1}{4})\|\mathbf{b}_j^\star\|^2$. On en déduit :

$$\text{si } i \leq j, \ \|\mathbf{b}_i^\star\|^2 \leq \alpha^{j-i}\|\mathbf{b}_j^\star\|^2. \tag{3.4}$$

En particulier : $\|\mathbf{b}_1\|^2 = \|\mathbf{b}_1^\star\|^2 \leq \alpha^{j-1}\|\mathbf{b}_j^\star\|^2$ pour tout j. On obtient en multipliant ces d inégalités, et en utilisant (3.1) :

$$\|\mathbf{b}_1\|^{2d} \leq \alpha^{\binom{n}{2}} \prod_{j=1}^{d} \|\mathbf{b}_j^\star\|^2 = \alpha^{\binom{n}{2}}(\mathrm{vol}L)^2.$$

Cela montre 1. Soit à présent un indice i. On a d'après (3.3) (réduction faible) et (3.4) : pour tout $j \geq i$,

$$\|\mathbf{b}_i\|^2 \leq \|\mathbf{b}_i^\star\|^2 + \frac{1}{4}\sum_{k=1}^{i-1}\|\mathbf{b}_k^\star\|^2 \leq \|\mathbf{b}_j^\star\|^2\left(\alpha^{j-i} + \frac{1}{4}\sum_{k=1}^{i-1}\alpha^{j-k}\right).$$

On va montrer que l'expression entre parenthèses est inférieure à α^{j-1}, ce qui équivaut à :

$$\alpha^{1-i} + \frac{1}{4}\sum_{k=1}^{i-1} \alpha^{1-k} \leq 1.$$

Cette inégalité est évidente pour $i = 1$. Pour $i \geq 2$, il vient en écrivant le terme de gauche en fonction de $\alpha^{-1} = \delta - \frac{1}{4} \leq \frac{3}{4}$:

$$\alpha^{1-i} + \frac{1}{4}\sum_{k=1}^{i-1} \alpha^{1-k} \leq \left(\frac{3}{4}\right)^{i-1} + \frac{1}{4} \times \frac{1 - (\frac{3}{4})^{i-1}}{1 - \frac{3}{4}} = \frac{1}{4} \times \frac{1}{1 - \frac{3}{4}} = 1.$$

On a donc pour tout $j \geq i$:

$$\|\mathbf{b}_i\|^2 \leq \alpha^{j-1}\|\mathbf{b}_j^\star\|^2.$$

Or d'après le lemme 3.3, il existe un indice $j \geq i$ tel que $\|\mathbf{b}_j^\star\| \leq \lambda_i(L)$. Comme $j \leq d$, on en déduit 2. Et on a aussi montré : $\|\mathbf{b}_i\|^2 \leq \alpha^{i-1}\|\mathbf{b}_i^\star\|^2$. Il vient en multipliant ces d inégalités :

$$\prod_{i=1}^{d} \|\mathbf{b}_i\|^2 \leq \alpha^{\binom{d}{2}} \prod_{i=1}^{d} \|\mathbf{b}_i^\star\|^2 = \alpha^{\binom{d}{2}} (\mathrm{vol} L)^2.$$

□

Ainsi, une base LLL-réduite fournit une solution approchée au problème de la réduction de réseaux. Notons que pour $\delta = \frac{3}{4}$, on obtient $\alpha = 2$. Les propriétés 1 et 3 sont en quelque sorte des versions algorithmiques des théorèmes de Minkowski, sur les minima successifs. Et la propriété 2 assure que chaque $\mathbf{b}_i$ n'est pas trop loin du minimum $\lambda_i(L)$. L'intérêt principal de cette notion de réduction provient du fait qu'il existe un algorithme simple qui permet de réduire une base au sens de LLL. Dans sa version la plus simple, l'algorithme LLL est celui décrit dans la figure 3.4. Il est d'autant plus intéressant qu'en pratique, les

Entrée : une base $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ d'un réseau L.
Sortie : la base $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ est LLL-réduite à un facteur δ.
1. Réduire faiblement $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ (voir figure 3.3).
2. S'il existe un indice j qui ne satisfait pas la condition de Lovász,
3. échanger $\mathbf{b}_j$ et $\mathbf{b}_{j+1}$, puis retourner à l'étape 1.

FIG. 3.4 – L'algorithme de réduction LLL.

bases renvoyées sont de bien meilleure qualité que ne le laisse supposer le théorème 3.11. En particulier, on constate expérimentalement que si le premier minimum est un peu plus petit que les autres minima, l'algorithme LLL trouve bien souvent un plus court vecteur. Et plus généralement, lorsque les premiers minima sont beaucoup plus faibles que les autres, il n'est pas rare que l'on retrouve le « sous-espace » engendré par ces premiers minima. L'algorithme LLL est en outre relativement efficace, puisqu'on a le résultat suivant :

Théorème 3.12 *On suppose ici $\frac{1}{4} < \delta < 1$. Si les $\mathbf{b}_i$ sont rationnels, l'algorithme LLL de la figure 3.4 fournit une base LLL-réduite à un facteur δ en temps polynomial en la taille maximale des coefficients des $\mathbf{b}_i$, la dimension du réseau et la dimension de l'espace.*

On va esquisser une preuve de ce résultat, en supposant pour simplifier les $\mathbf{b}_i$ entiers. Tout d'abord, il est clair que si l'algorithme termine, la base obtenue est LLL-réduite pour le facteur δ. Pour voir que l'algorithme termine, analysons chaque échange (étape 3). Lorsqu'on échange $\mathbf{b}_j$ et $\mathbf{b}_{j+1}$, seuls $\mathbf{b}_j^\star$ et $\mathbf{b}_{j+1}^\star$ peuvent être modifiés parmi les vecteurs de Gram-Schmidt. Notons donc $\mathbf{c}_j^\star$ et $\mathbf{c}_{j+1}^\star$ les nouveaux vecteurs après échange. Comme le produit des normes des vecteurs de Gram-Schmidt vaut vol$L$, on a :

$$\|\mathbf{c}_j^\star\| \times \|\mathbf{c}_{j+1}^\star\| = \|\mathbf{b}_j^\star\| \times \|\mathbf{b}_{j+1}^\star\|.$$

Le fait que la condition de Lovász ne soit pas satisfaite s'écrit : $\|\mathbf{c}_j^\star\|^2 < \delta\|\mathbf{b}_j^\star\|^2$. On en déduit :

$$\|\mathbf{c}_j^\star\|^{2(d-j+1)}\|\mathbf{c}_{j+1}^\star\|^{2(d-j)} < \delta\|\mathbf{b}_j^\star\|^{2(d-j+1)}\|\mathbf{b}_{j+1}^\star\|^{2(d-j)}.$$

Cela nous invite à considérer la quantité :

$$D = \|\mathbf{b}_1^\star\|^{2d}\|\mathbf{b}_2^\star\|^{2(d-1)} \times \cdots \times \|\mathbf{b}_d^\star\|^d.$$

À chaque échange, D diminue d'un facteur $\delta < 1$. On remarque que D peut se décomposer comme un produit de d déterminants de Gram $D_i = \Delta(\mathbf{b}_1, \ldots, \mathbf{b}_i)$ pour i variant de 1 à d. D est donc en fait un entier, puisque les $\mathbf{b}_i$ le sont. Il s'en suit que le nombre d'échanges est au plus logarithmique en la valeur initiale de D, que l'on peut majorer par B^{2n} où B est le maximum des normes initiales $\|\mathbf{b}_i\|$. Pour borner la complexité de l'algorithme, il faut aussi majorer la taille des coefficients rationnels $\mu_{i,j}$. Une analyse minutieuse fondée sur les D_i permet de montrer que les $\mu_{i,j}$ restent polynomiaux. Mais on ne sait pas si l'algorithme reste polynomial pour $\delta = 1$, si bien qu'en pratique, on choisit généralement $\delta = 0.99$.

L'algorithme décrit en figure 3.4 est loin d'être optimal. On peut notamment réécrire l'algorithme de façon à ne réduire faiblement qu'une partie de la base entre chaque échange. Et l'on peut minimiser les calculs de coefficients de Gram-Schmidt. Pour des versions « optimisées » de l'algorithme LLL, la complexité est $\mathcal{O}(nd^5m^3)$, m étant le nombre maximal de bits des coordonnées des $\mathbf{b}_i$, n la dimension de l'espace, et d la dimension du réseau. Cette borne supérieure s'avère néanmoins pessimiste en pratique.

Chapitre 4 (6h)

La sécurité prouvée en chiffrement asymétrique

— par David Pointcheval

Sommaire

4.1 La cryptographie asymétrique 72
4.1.1 Limites de la cryptographie conventionnelle 72
4.1.2 Chiffrement asymétrique 72
4.1.3 Le cryptosystème RSA 72
Présentation 73
Sécurité du chiffrement RSA 73
4.2 Formalisation 73
4.2.1 Chiffrement à clé publique 73
4.2.2 Notions de sécurité 74
Buts d'un attaquant 74
Les moyens d'un attaquant 75
Quantification de la sécurité 75
4.2.3 Relations entre les notions de sécurité 76
4.3 Les cryptosystèmes RSA et Rabin 78
4.3.1 Le chiffrement RSA 78
4.3.2 Le chiffrement de Rabin 79
Description 79
Résultats de sécurité 79
4.4 Le problème du logarithme discret 80
4.4.1 Énoncé des problèmes 81
4.4.2 Difficulté du logarithme discret 81
4.5 Le cryptosystème de El Gamal 82
4.5.1 Description 82
4.5.2 Résultats de sécurité 82
4.6 Les attaques à chiffrés choisis adaptatives 84
4.6.1 Le modèle de l'oracle aléatoire 84
4.6.2 Première construction générique 84
4.6.3 Constructions plus génériques 86

4.1 La cryptographie asymétrique

4.1.1 Limites de la cryptographie conventionnelle

L'inconvénient majeur de la cryptographie conventionnelle est la nécessité de partager une clé secrète entre tout couple de participants. Ainsi, un ensemble de n personnes fait intervenir n^2 clés !

De plus, cela sous-entend qu'avant toute communication, deux interlocuteurs doivent avoir mis en commun un secret connu d'eux seuls. Comment est-ce possible, s'ils sont éloignés de plusieurs milliers de kilomètres ? Il faut avoir un canal sécurisé pour établir une clé commune, en vue d'un canal sécurisé . . . !

Pourtant, si on pousse les principes de Kerckhoffs à leur maximum (dont l'un dit que la sécurité des protocoles ne doit pas reposer sur leur caractère secret), en quoi la fonction de chiffrement devrait-elle être « secrète », ou pourquoi devrait-elle faire appel à une clé secrète ? Ce que l'on souhaite d'un schéma de chiffrement est qu'il empêche un intrus de décrypter le message chiffré, non de chiffrer.

4.1.2 Chiffrement asymétrique

Dans un système de chiffrement asymétrique, chaque utilisateur possède deux clés, l'une privée (notée sk), l'autre publique (notée pk). Lorsqu'Alice souhaite envoyer un message chiffré pour Bob, elle utilise l'algorithme de chiffrement $\mathcal{E}$ avec la clé publique pk de Bob pour produire le chiffré $c = \mathcal{E}_{\mathsf{pk}}(m)$. À la réception de c, Bob peut utiliser l'algorithme de déchiffrement $\mathcal{D}$ avec sa clé privée sk pour retrouver le message initial (voir la figure 4.1).

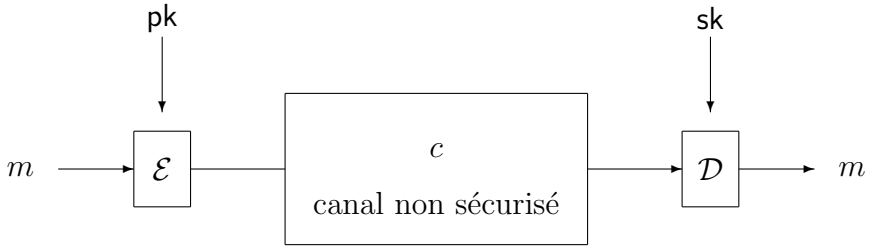


Fig. 4.1 – Chiffrement asymétrique

En pratique, ces algorithmes $\mathcal{E}$ et $\mathcal{D}$ doivent avoir les propriétés suivantes :
- pour tout message m, $\mathcal{D}_{\mathsf{sk}}(\mathcal{E}_{\mathsf{pk}}(m)) = m$,
- retrouver m à partir de $\mathcal{E}_{\mathsf{pk}}(m)$ doit être calculatoirement impossible, à moins de connaître la clé privée sk.

On dit alors que la fonction $\mathcal{E}$ est *à sens unique* (car calculatoirement non-inversible) mais *à trappe* (car sk rend cette fonction aisément inversible).

4.1.3 Le cryptosystème RSA

Comme il a été vu dans la section 1.4.2, le système RSA propose une telle famille de fonctions (et même de permutations) à sens-unique. La factorisation du module RSA fournissant une trappe.

Soient n un entier RSA (de la forme $n = pq$)
et e un exposant premier avec $\varphi(n)$.
- Clé publique d'Alice : $\mathsf{pk} = (n, e)$
- Clé privée d'Alice : $\mathsf{sk} = d = e^{-1} \bmod \varphi(n)$
- Chiffrement : soit m un message à chiffrer pour Alice.
  Le message m est vu comme un élément de $\mathbb{Z}_n^\star$,
  et son chiffré est alors $c = \mathcal{E}_{\mathsf{pk}}(m) = m^e \bmod n$
- Déchiffrement : seule Alice est capable de retrouver m
  à partir de c et de d, $m = \mathcal{D}_{\mathsf{sk}}(c) = c^d \bmod n$

FIG. 4.2 – Chiffrement RSA

Définition 4.1 (Le problème RSA) ou la racine e-ième modulaire. *Soient $n = pq$ un entier, e un exposant premier avec $\varphi(n)$ et $x \in \mathbb{Z}_n^\star$, trouver une racine e-ième de x modulo n, soit un élément $y \in \mathbb{Z}_n^\star$ tel que $y^e = x \bmod n$.*

Nous avons vu un moyen de résoudre ce problème, avec la factorisation de n, ou de façon équivalente la valeur de $\varphi(n)$:

$$(x^d)^e = x^{d \cdot e} = x \bmod n, \text{ si } d = e^{-1} \bmod \varphi(n).$$

Présentation

Comme nous venons de le constater, le problème RSA présente les propriétés requises pour du chiffrement asymétrique, il propose un candidat comme fonction à sens-unique à trappe : le calcul de racines e-ième est calculatoirement impossible (hypothèse RSA) à moins de connaître la factorisation de n.

Nous pouvons donc définir plus formellement le premier algorithme de chiffrement asymétrique (proposé en 1978 par R. Rivest, A. Shamir et L. Adleman [18]) connu sous le nom de RSA (voir la figure 4.2).

Sécurité du chiffrement RSA

Théorème 4.1 *L'inversion du chiffrement RSA est équivalente au problème RSA, lui-même supposé aussi difficile que la factorisation.*

En pratique, jusqu'à présent, des entiers n de 512 bits étaient utilisés (soit 155 chiffres), mais depuis le récent record de factorisation, il est préconisé d'utiliser des entiers de 768 bits ou 1024 bits.

4.2 Formalisation

4.2.1 Chiffrement à clé publique

Le but du chiffrement à clé publique est de permettre à quiconque connaissant la clé publique d'Alice de lui envoyer un message qu'elle seule sera en mesure de lire, grâce à sa clé privée.

Un schéma de chiffrement à clé publique est défini par les trois algorithmes suivants :

– L'*algorithme de génération des clés* $\mathcal{K}$. En fonction d'un paramètre de sécurité k, l'algorithme $\mathcal{K}(1^k)$ retourne une paire de clés publique/privée associées $(\mathsf{pk}, \mathsf{sk})$. Cet algorithme $\mathcal{K}$ est probabiliste.
– L'*algorithme de chiffrement* $\mathcal{E}$. Étant donné un message $m \in \mathcal{M}$ et une clé publique pk, $\mathcal{E}_{\mathsf{pk}}(m)$ produit un chiffré c de m. Cet algorithme peut être probabiliste. Dans ce cas, on utilisera la notation $\mathcal{E}_{\mathsf{pk}}(m, r)$, où r est l'aléa fourni à l'algorithme $\mathcal{E}$.
– L'*algorithme de déchiffrement* $\mathcal{D}$. Étant donné un chiffré c et la clé privée sk (associée à pk), $\mathcal{D}_{\mathsf{sk}}(c)$ retourne le message clair m correspondant, ou $\perp$ pour un chiffré non valide. Cet algorithme est nécessairement déterministe.

4.2.2 Notions de sécurité

Afin d'évaluer la sécurité effective d'un schéma de chiffrement, il faut formaliser les notions que l'on souhaite garantir. On précise alors les *buts* qu'un attaquant peut vouloir atteindre, et que l'on veut éviter. De façon orthogonale, on explicite les informations accessibles à l'attaquant, les *moyens* qu'il peut mettre en œuvre.

Buts d'un attaquant

Comme on l'a vu pour le chiffrement RSA, l'objectif majeur d'un attaquant est bien sûr de retrouver l'intégralité du message clair, à partir du seul chiffré, et des informations publiques.

La formalisation de cette notion de sécurité est le fait, pour la fonction de chiffrement, d'être *à sens unique* (ou *one-wayness* – OW) : pour tout adversaire $\mathcal{A}$, sa probabilité de succès à inverser $\mathcal{E}_{\mathsf{pk}}$ sans la clé privée sk est négligeable sur l'espace de probabilité $\mathcal{M} \times \Omega$, où $\mathcal{M}$ est l'espace des messages clairs et Ω l'espace des aléas (dans le cas d'un algorithme probabiliste), ainsi que sur le ruban aléatoire de l'attaquant $\mathcal{A}$:

$$\mathsf{Succ}^{\mathsf{ow}}(\mathcal{A}) = \Pr_{m,r}\left[(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathcal{K}(1^k) : \mathcal{A}(\mathsf{pk}, \mathcal{E}_{\mathsf{pk}}(m, r)) = m\right].$$

Cependant, l'attaquant peut se contenter d'une information partielle sur le message clair. Malheureusement, une sécurité parfaite est impossible (voir section 1.2). En effet, la distribution *a posteriori* du message clair, avec la vue du chiffré et de la clé publique, est réduite à un point. Cependant, on peut définir une version calculatoire de la sécurité parfaite. En effet, une autre formulation de la sécurité parfaite est qu'un attaquant tout puissant ne peut prédire un seul bit du message clair. La *sécurité sémantique* (ou *sécurité polynomiale* [11], et encore l'*indistinguabilité des chiffrés* – IND) se contente de considérer les attaquants polynomiaux. Formellement, cela se traduit par l'impossibilité pour tout attaquant de distinguer, parmi deux messages de son choix, lequel est chiffré dans le cryptogramme donné, ou *challenge*.

Avec un choix aléatoire, tout attaquant peut « gagner »avec probabilité 1/2. Ainsi on considère l'avantage qu'un attaquant peut avoir par rapport à un « lancer de pièce » :

$$\begin{aligned}\mathsf{Adv}^{\mathsf{ind}}(\mathcal{A}) &= \left| 2 \times \Pr_{b,r}\left[\begin{array}{l}(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathcal{K}(1^k), (m_0, m_1, s) \leftarrow A_1(\mathsf{pk}), \\ c = \mathcal{E}_{\mathsf{pk}}(m_b, r), b' = A_2(m_0, m_1, s, c) : b' = b\end{array}\right] - 1 \right| \\ &= \left| \Pr_r[b' = 1 \,|\, b = 1] - \Pr_r[b' = 1 \,|\, b = 0] \right|,\end{aligned}$$

où l'attaquant $\mathcal{A}$ fonctionne en deux temps, (A_1, A_2) : dans un premier temps, à la vue de la clé publique, l'algorithme A_1 choisit deux messages de même taille pour lesquels il estime

qu'il saura distinguer les chiffrés. Ce que tente de faire l'algorithme A_2 sur le challenge c. La variable s permet seulement à A_1 de transmettre formellement de l'information à la deuxième étape A_2.

Une autre notion s'est ensuite révélée utile, la *non-malléabilité* (NM) [9]. Cette notion consiste à empêcher un attaquant, étant donné un chiffré $c = \mathcal{E}_{\mathsf{pk}}(m)$, de produire un nouveau chiffré $c^\star = \mathcal{E}_{\mathsf{pk}}(m^\star)$ tels que les messages m et $m^\star$ satisfassent une relation particulière. Pour formaliser cette notion, on considère à nouveau un attaquant $\mathcal{A} = (A_1, A_2)$ en deux étapes. Dans un premier temps, l'algorithme A_1, à la vue de la clé publique pk, retourne une distribution sur l'ensemble des messages, caractérisée par un algorithme d'échantillonnage M. Un tel algorithme ne doit retourner avec une probabilité non nulle que des messages de même taille ; Dans un deuxième temps, l'algorithme A_2 reçoit le chiffré y d'un message aléatoire x (selon la distribution M). Cet adversaire retourne une relation R et un vecteur $\mathbf{y}$ de chiffrés (tous différents de y). Il espère que $R(x, \mathbf{x})$ soit satisfaite, où $\mathbf{x}$ est le déchiffrement de $\mathbf{y}$, coordonnée par coordonnée.

Un tel attaquant réussit dans son attaque s'il parvient à satisfaire la relation ci-dessus avec une meilleure probabilité que sur un message aléatoire inconnu : $R(x^\star, \mathbf{x})$ avec $x^\star \leftarrow M$.

$$\mathsf{Adv}^{\mathsf{nm}}(\mathcal{A}) = \left|\mathsf{Succ}^{M}(\mathcal{A}) - \mathsf{Succ}^{\$}(\mathcal{A})\right|, \text{ avec}$$

$$\left.\begin{array}{rcl} \mathsf{Succ}^{M}(\mathcal{A}) & = & \Pr\left[\begin{array}{c} y \notin \mathbf{y} \ \wedge \ \perp \notin \mathbf{x} \\ \wedge\ R(x, \mathbf{x}) \end{array}\right] \\ \mathsf{Succ}^{\$}(\mathcal{A}) & = & \Pr\left[\begin{array}{c} y \notin \mathbf{y} \ \wedge \ \perp \notin \mathbf{x} \\ \wedge\ R(x^\star, \mathbf{x}) \end{array}\right] \end{array}\right\} \begin{array}{c} \text{sur l'espace de probabilités défini par} \\ \\ (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathcal{K}(1^k), (M, s) \leftarrow A_1(\mathsf{pk}), \\ x, x^\star \leftarrow M, y = \mathcal{E}_{\mathsf{pk}}(x, r), \\ (R, \mathbf{y}) \leftarrow A_2(M, s, y), \mathbf{x} = \mathcal{D}_{\mathsf{sk}}(\mathbf{y}). \end{array}$$

Les moyens d'un attaquant

Dans le contexte asymétrique, grâce à la clé publique, un attaquant peut chiffrer tout message de son choix. Il peut donc mettre en œuvre l'attaque de base appelée *à clairs choisis* (ou *chosen-plaintext attack* – CPA).

Mais cet attaquant peut avoir accès à plus d'information, notamment apprendre, pour tout chiffré y de son choix, s'il est valide ou non : $\mathcal{D}_{\mathsf{sk}}(y) \stackrel{?}{=} \perp$. On parle d'attaque *avec test de validité* (ou *validity checking attack* – VCA) [7].

On peut également imaginer l'accès, pour tout couple (x, y), à l'information de relation : $\mathcal{D}_{\mathsf{sk}}(y) \stackrel{?}{=} x$. On parle d'attaque *avec vérification du clair* (ou *plaintext checking attack* – PCA) [14].

Dans certains cas, l'attaquant peut même avoir accès à l'algorithme de déchiffrement. Si cet accès n'est possible qu'avant la vue du challenge (dans la première étape de l'attaque), on parle d'attaque *à chiffrés choisis non-adaptative* (ou *non-adaptive chosen-ciphertext attack* – CCA1) [13]. Si cet accès est illimité (avec la restriction naturelle de ne pas l'utiliser sur le challenge), il s'agit d'attaques *à chiffrés choisis adaptatives* (ou *adaptive chosen-ciphertext attack* – CCA2) [17].

Quantification de la sécurité

Pour définir plus précisément le niveau de sécurité, on note $\mathsf{Succ}^{\mathsf{xxx}}(t(k))$ ou $\mathsf{Adv}^{\mathsf{xxx}}(t(k))$ la probabilité, respectivement succès ou avantage, maximale qu'un attaquant de type xxx (définissant l'objectif et les moyens, par exemple ind-cpa) en temps $t(k)$, où k est le paramètre de sécurité. Ce dernier sera par la suite omis mais sous-entendu.

Si pour un système donné, ε est supérieure à $\mathsf{Succ}^{\mathsf{xxx}}(t)$ ou $\mathsf{Adv}^{\mathsf{xxx}}(t)$, alors on dit que ce système est (t,ε)–XXX-sûr. On quantifiera de la même manière la difficulté d'un problème calculatoire : $\mathsf{Succ}^{\mathsf{rsa}}(t)$ dénote le succès maximal de tout attaquant en temps t contre le problème RSA (pour des modules de k-bits).

4.2.3 Relations entre les notions de sécurité

Les attaques essentielles sont, l'attaque de base à clairs choisis (CPA), puis les plus puissantes à chiffrés choisis (CCA1 et CCA2), d'où les relations partielles présentées sur la figure 4.3 : la non-malléabilité entraîne la sécurité sémantique, quel que soit le type d'attaque. En revanche, dans le scénario des attaques à chiffrés choisis adaptatives, ces deux notions sont équivalentes. On parle alors de sécurité *face aux attaques à chiffrés choisis.*

Théorème 4.2 *[NM⟹IND].* *Si le schéma de chiffrement $(\mathcal{K},\mathcal{E},\mathcal{D})$ est non-malléable, alors il est sémantiquement sûr, selon le même type d'attaque :*

$$\mathsf{Adv}^{\mathsf{ind-atk}}(t) \leq 2 \times \mathsf{Adv}^{\mathsf{nm-atk}}(t + T_{\mathcal{E}}),$$

où $T_{\mathcal{E}}$ désigne le temps nécessaire pour un chiffrement.

Preuve. Nous supposons que le schéma $(\mathcal{K},\mathcal{E},\mathcal{D})$ est non-malléable. Nous allons montrer qu'il est alors sémantiquement sûr. Pour cela, considérons un attaquant $\mathcal{B} = (B_1, B_2)$ contre la sécurité sémantique, nous allons montrer que $\mathsf{Adv}^{\mathsf{ind}}(\mathcal{B})$ est faible.

Construisons donc un adversaire $\mathcal{A} = (A_1, A_2)$ contre la non-malléabilité, qui a accès aux mêmes oracles que $\mathcal{B}$:

Algorithme $A_1(\mathsf{pk})$	Algorithme $A_2(M, s', y)$ où $s' = (x_0, x_1, \mathsf{pk}, s)$
$(x_0, x_1, s) \leftarrow B_1(\mathsf{pk})$	$d \leftarrow B_2(x_0, x_1, s, y)$
$M := \{x_0, x_1\}$	$y' \leftarrow \mathcal{E}_{\mathsf{pk}}(\overline{x_d})$
$s' \leftarrow (x_0, x_1, \mathsf{pk}, s)$	$\mathsf{Return}(R, y')$ où $R(a,b) = 1$ ssi $a = \overline{b}$
$\mathsf{Return}(M, s')$	

Le mot $\overline{b}$ désigne b où tous les bits sont inversés. La notation $M := \{x_0, x_1\}$ signifie que la distribution produite par M retourne x_0 ou x_1, avec une probabilité identique (soit 1/2). En effet, nous supposons que le couple (x_0, x_1) retourné par B_1 est toujours constitué de deux messages distincts, puisque la contribution à l'avantage sur les exécutions où $x_0 = x_1$ est parfaitement nulle. On pourrait donc modifier l'attaquant $\mathcal{B}$ sur ces exécutions, en lui faisant retourner deux messages différents, et en choisissant d aléatoirement, sans dégrader l'avantage. On suppose donc par la suite que $x_0 \neq x_1$.

A_2 retourne (la description de) une relation R, qui pour toute entrée (a,b) est satisfaite (vaut 1) si et seulement si $a = \overline{b}$, et n'est pas satisfaite (vaut 0) dans les autres cas.

Considérons l'avantage de l'attaquant $\mathcal{A}$ contre la non-malléabilité du système :

$$\mathsf{Adv}^{\mathsf{nm}}(\mathcal{A}) = \left|\mathsf{Succ}^{M}(\mathcal{A}) - \mathsf{Succ}^{\$}(\mathcal{A})\right|, \text{ avec}$$

$$\left.\begin{array}{rcl} \mathsf{Succ}^{M}(\mathcal{A}) & = & \Pr\left[\begin{array}{c} y' \neq y \ \wedge \ x' \neq \bot \\ \wedge \ R(x,x') \end{array}\right] \\ \mathsf{Succ}^{\$}(\mathcal{A}) & = & \Pr\left[\begin{array}{c} y' \neq y \ \wedge \ x' \neq \bot \\ \wedge \ R(x^\star,x') \end{array}\right] \end{array}\right\} \begin{array}{l} \text{sur l'espace de probabilités défini par} \\ \\ (\mathsf{pk},\mathsf{sk}) \leftarrow \mathcal{K}(1^k), (M,s) \leftarrow A_1(\mathsf{pk}), \\ x, x^\star \leftarrow M, y = \mathcal{E}_{\mathsf{pk}}(x,r), \\ (R,y') \leftarrow A_2(M,s,y), x' = \mathcal{D}_{\mathsf{sk}}(y'). \end{array}$$

Rappelons que l'avantage de $\mathcal{B}$ contre la sécurité sémantique est $\mathsf{Adv}^{\mathsf{ind}}(\mathcal{B}) = |\, 2 \times p_k - 1 \,|$, où

$$p_k = \Pr_{b,r}\left[\begin{array}{l}(\mathsf{pk},\mathsf{sk}) \leftarrow \mathcal{K}(1^k), (x_0, x_1, s) \leftarrow B_1(\mathsf{pk}),\\ y = \mathcal{E}_{\mathsf{pk}}(x_b, r), d \leftarrow B_2(x_0, x_1, s, c) : d = b\end{array}\right].$$

Puis évaluons successivement $\mathsf{Succ}^M(\mathcal{A})$ et $\mathsf{Succ}^{\$}(\mathcal{A})$.

Lemme 4.1 $\mathsf{Succ}^M(\mathcal{A}) = p_k$.

Preuve. (du lemme). Si on regarde le fonctionnement de A_2, on constate que $R(x, x')$ est vrai si et seulement si $\mathcal{D}_{\mathsf{sk}}(y) = x_d$. Remarquons également que lorsque $R(x, x')$ est vrai, nous avons nécessairement $x \neq x'$ et, par unicité du déchiffré, $y \neq y'$. De plus, nous avons toujours $x' \neq \perp$.

Ré-écrivons désormais la définition de $\mathsf{Succ}^M(\mathcal{A})$, il s'agit de

$$\mathsf{Succ}^M(\mathcal{A}) = \Pr\left[\begin{array}{c}(\mathsf{pk},\mathsf{sk}) \leftarrow \mathcal{K}(1^k), (x_0, x_1, s) \leftarrow B_1(\mathsf{pk}), b \xleftarrow{R} \{0,1\},\\ x = x_b, y = \mathcal{E}_{\mathsf{pk}}(x, r), d \leftarrow B_2(x_0, x_1, s, y), x' = \overline{x_d} : b = d\end{array}\right] = p_k.$$

□

Lemme 4.2 $\mathsf{Succ}^{\$}(\mathcal{A}) = 1/2$.

Preuve. (du lemme). Ceci provient simplement du fait que l'attaquant n'a aucune information (même avec une puissance de calcul infinie) sur le message $x^\star$ auquel va être comparé x'. Il peut s'agir de x_0 ou de x_1 avec une distribution uniforme. □

On peut alors combiner les deux lemmes pour obtenir

$$\mathsf{Adv}^{\mathsf{ind}}(\mathcal{B}) = 2 \cdot \left| p_k - \frac{1}{2} \right| = 2 \cdot \Big|\, \mathsf{Succ}^M(\mathcal{A}) - \mathsf{Succ}^{\$}(\mathcal{A}) \,\Big| = 2 \cdot \mathsf{Adv}^{\mathsf{nm}}(\mathcal{A}).$$

□

D'un autre côté ces deux notions sont distinctes (on peut exhiber des schémas qui atteignent l'une des notions mais pas l'autre) contre des attaques CPA et CCA1. En revanche, on peut montrer

Théorème 4.3 *[IND-CCA2 $\Longrightarrow$ NM-CCA2].* *Si le schéma de chiffrement $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ est sémantiquement sûr contre des attaques à chiffrés choisis adaptatives alors il est non-malléable selon cette même attaque :*

$$\mathsf{Adv}^{\mathsf{nm-cca2}}(t) \leq 2 \times \mathsf{Adv}^{\mathsf{ind-cca2}}(t + T_R + \mathcal{O}(1)),$$

où T_R désigne le temps nécessaire pour évaluer la relation R.

Preuve. Nous supposons un schéma de chiffrement $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ qui soit sémantiquement sûr contre des attaques à chiffrés choisis adaptatives. Nous allons montrer qu'il est également non-malléable.

Nous considérons un attaquant $\mathcal{B} = (B_1^{\mathcal{D}_{\mathsf{sk}}}, B_2^{\mathcal{D}_{\mathsf{sk}}})$ contre la non-malléabilité. Les deux étapes B_1 et B_2 ont accès à l'algorithme de déchiffrement, d'où la notation avec des oracles. Nous allons montrer que $\mathsf{Adv}^{\mathsf{nm}}(\mathcal{B})$ est négligeable. Pour cela, comme précédemment, nous construisons un adversaire $\mathcal{A} = (A_1^{\mathcal{D}_{\mathsf{sk}}}, A_2^{\mathcal{D}_{\mathsf{sk}}})$ contre la sécurité sémantique.

Algorithme $A_1^{\mathcal{D}_{\mathsf{sk}}}(\mathsf{pk})$	Algorithme $A_2^{\mathcal{D}_{\mathsf{sk}}}(x_0, x_1, s', y)$ où $s' = (M, s)$
$(M, s) \leftarrow B_1^{\mathcal{D}_{\mathsf{sk}}}(\mathsf{pk})$	$(R, \mathbf{y}) \leftarrow B_2^{\mathcal{D}_{\mathsf{sk}}}(M, s, y)$, $\mathbf{x} \leftarrow \mathcal{D}_{\mathsf{sk}}(\mathbf{y})$
$x_0 \leftarrow M$, $x_1 \leftarrow M$	if $(y \notin \mathbf{y} \ \wedge \ \perp \notin \mathbf{x} \ \wedge R(x_0, \mathbf{x}))$ then $d \leftarrow 0$
$s' := (M, s)$	else $d \leftarrow \{0, 1\}$
Return (x_0, x_1, s')	Return d

Pour cette réduction, nous voyons qu'il est important que l'évaluation de R soit polynomiale, ainsi que le tirage d'un élément selon la distribution M. L'avantage de notre attaquant $\mathcal{A}$ est $\mathsf{Adv}^{\mathsf{ind}}(\mathcal{A}) = |\, p_k(0) - p_k(1)\,|$ où, pour $b \in \{0, 1\}$ on définit

$$p_k(b) = \Pr\left[\begin{array}{l}(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathcal{K}(1^k), (x_0, x_1, s) \leftarrow A_1^{\mathcal{D}_{\mathsf{sk}}}(\mathsf{pk}), \\ c = \mathcal{E}_{\mathsf{pk}}(x_b) : A_2^{\mathcal{D}_{\mathsf{sk}}}(x_0, x_1, s, c) = 0\end{array}\right].$$

Aussi, pour $b \in \{0, 1\}$, on définit

$$p'_k(b) = \Pr\left[\begin{array}{c}(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathcal{K}(1^k), (M, s) \leftarrow B_1^{\mathcal{D}_{\mathsf{sk}}}(\mathsf{pk}), x_0, x_1 \leftarrow M, \\ c = \mathcal{E}_{\mathsf{pk}}(x_b), (R, \mathbf{y}) \leftarrow B_2^{\mathcal{D}_{\mathsf{sk}}}(M, s, c), \mathbf{x} = \mathcal{D}_{\mathsf{sk}}(\mathbf{y}) : \\ y \notin \mathbf{y} \ \wedge \ \perp \notin \mathbf{x} \ \wedge \ R(x_0, \mathbf{x})\end{array}\right]$$

On remarque que A_2 peut retourner 0, soit parce que $\mathbf{x}$ est en relation avec x_0, soit par tirage au sort. Alors,

$$\mathsf{Adv}^{\mathsf{ind}}(\mathcal{A}) = |\, p_k(0) - p_k(1)\,| = \left|\frac{1}{2} \cdot [1 + p'_k(0)] - \frac{1}{2} \cdot [1 + p'_k(1)]\right| = \frac{1}{2} \cdot |\, p'_k(0) - p'_k(1)\,|.$$

On peut aussi remarquer que l'exécution de B_2, recevant un chiffré de x_1, retournant un $\mathbf{y}$ tel que $\mathbf{x}$ est en relation avec x_0 définit exactement $\mathsf{Succ}^{\$}(\mathcal{B})$. D'un autre côté, s'il a reçu un chiffré de x_0, cela définit $\mathsf{Succ}^{M}(\mathcal{B})$. Par conséquent,

$$\mathsf{Adv}^{\mathsf{nm}}(\mathcal{B}) = p'_k(0) - p'_k(1) = 2 \cdot \mathsf{Adv}^{\mathsf{ind}}(\mathcal{A}).$$

□

Théorème 4.4

$$\mathit{IND-CCA2} \Longleftrightarrow \mathit{NM-CCA2}.$$

Une étude plus complète des relations entre les différentes notions de sécurité a été menée dans [3]. Elle présente les preuves des implications présentées sur la figure 4.3. La *one-wayness* n'apparaît pas sur ce schéma car cette notion de sécurité s'est révélée peu robuste : on a récemment montré que les propriétés de *sécurité sémantique* et de *non-malléabilité* étaient conservées même si l'on chiffrait le message sous plusieurs clés simultanément [1, 2], ça n'est pas le cas de la *one-wayness*, comme le suggère le contre-exemple de RSA, avec la célèbre attaque par broadcast [12].

Ainsi cherche-t-on à construire des schémas de chiffrement qui atteignent la notion de sécurité maximale, contre les attaques à chiffrés choisis.

4.3 Les cryptosystèmes RSA et Rabin

4.3.1 Le chiffrement RSA

Le chiffrement RSA présenté figure 4.2 apporte donc un niveau de sécurité minimal : il est OW-CPA, sous l'hypothèse RSA, ou plus précisément, $\mathsf{Succ}^{\mathsf{ow-cpa}}(t) \leq \mathsf{Succ}^{\mathsf{rsa}}(t)$. Il n'y a aucune chance d'obtenir un niveau de sécurité plus important :

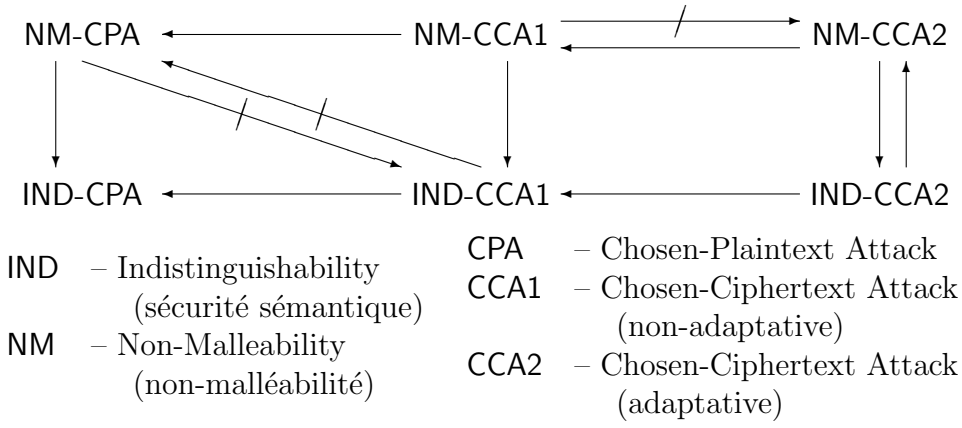


FIG. 4.3 – Relations entre les notions de sécurité

- la sécurité sémantique est exclue en raison du déterminisme de l'algorithme de chiffrement ;
- les attaques à chiffrés choisis adaptatives permettent d'inverser le chiffrement en tout point, à cause de la propriété homomorphique ;
- les oracles de validité —VCA— ou de vérification du clair —PCA— sont sans intérêt.

Le niveau de sécurité OW-CCA1 est cependant ouvert. Il ne repose pas sur l'hypothèse RSA, mais sur la difficulté du « one-more RSA » : est-ce qu'un oracle RSA, accessible momentanément, pourrait nous servir à résoudre une instance donnée ultérieurement ?

Une hypothèse similaire a été faite récemment [4], mais c'est une hypothèse plus forte que la seule hypothèse RSA. En revanche, l'équivalence entre le problème RSA et la factorisation contre-dirait cette dernière hypothèse.

Théorème 4.5 *Si le problème de la factorisation est équivalent au problème RSA, alors le problème du « one-more RSA »est facile.*

Preuve. En effet, une telle équivalence signifie qu'un oracle RSA nous permet de factoriser le module. On peut alors ensuite calculer l'exposant de déchiffrement et inverser toute instance ultérieure. □

4.3.2 Le chiffrement de Rabin

Description

Michael O. Rabin a proposé peu après un cryptosystème basé sur le problème de la racine carrée modulaire, et donc équivalent à la factorisation (voir figure 4.4). Cependant, cette équivalence est conditionnée par de sévères critères qui peuvent sembler contradictoires avec les aspects pratiques : la redondance ne doit pas être trop stricte, sinon la preuve s'effondre ; en revanche, si elle ne l'est pas assez, on risque d'avoir plusieurs racines valides pour un seul chiffré. Une dizaine de bits de redondance semble convenable.

Résultats de sécurité

Théorème 4.6 *Avec quelques bits de redondance, le chiffrement de Rabin est OW-CPA sous réserve de la difficulté de la factorisation.*

Soit $n = pq$ un entier RSA.
- Clé publique d'Alice : n et $B \in \mathbb{Z}_n$
- Clé privée d'Alice : p, q
- Chiffrement : soit m un message à chiffrer pour Alice, vu comme un élément de $\mathbb{Z}_n$, son chiffré est alors $c = m(m + B) \bmod n$
- Déchiffrement : seule Alice est capable de retrouver m à partir de c, car cela nécessite le calcul des racines carrées modulo n de $c + B^2/4$, puis d'y retrancher $B/2$:

$$m \in \{M_1, M_2, M_3, M_4\} = \sqrt{c + \frac{B^2}{4}} - \frac{B}{2} \bmod n.$$

Pour déterminer m parmi les quatre messages possibles, il faut lui introduire quelques bits de redondance.

FIG. 4.4 – Chiffrement de Rabin

Preuve. Supposons que la redondance porte sur ℓ bits. On choisit $x \in \mathbb{Z}_n$, puis on calcule le chiffré de x, $c = x(x + B) \bmod n$. On donne ce chiffré à l'attaquant. Il s'agit d'un chiffré valide (avec un clair qui satisfait la redondance souhaitée) avec probabilité supérieure à $1/2^\ell$ (de l'ordre de $4/2^\ell$). Dans ce cas, l'attaquant nous retourne le clair m avec probabilité ε :

$$m(m + B) = \left(m - \frac{B}{2}\right)^2 - \frac{B^2}{4} = \left(x - \frac{B}{2}\right)^2 - \frac{B^2}{4} \bmod n.$$

Ainsi, avec probabilité $1/2$, le `pgcd` de $x - m$ et n nous fournit un facteur premier de n. La probabilité de succès de cette réduction est alors supérieure à $\varepsilon/2^{\ell+1}$: la redondance ne doit donc pas être trop importante. □

Cependant, comme pour RSA, et même de façon plus flagrante, il n'y a aucune chance d'obtenir un niveau de sécurité plus important :
- la sécurité sémantique est exclue en raison du déterminisme de l'algorithme de chiffrement ;
- et l'oracle de déchiffrement, et donc les attaques à chiffrés choisis, adaptatives ou non, permet de retrouver la clé privée, et donc d'inverser ultérieurement tout chiffré.
- l'oracle de vérification du clair —PCA— est sans intérêt ;

Selon la redondance, l'oracle de validité d'un chiffré peut éventuellement permettre de retrouver la clé privée, notamment si la redondance consiste en quelques bit de poids faible ou de poids fort mis à zéro [7].

4.4 Le problème du logarithme discret

Le problème RSA repose sur la difficulté de déterminer l'ordre d'un groupe (en l'occurrence le sous-groupe multiplicatif de $\mathbb{Z}_n$, pour n composé. Le problème du logarithme discret se pose même lorsque l'on connaît cet ordre.

4.4.1 Énoncé des problèmes

Ainsi, considérons un groupe cyclique fini $\mathcal{G}$, d'ordre q, (on le considèrera premier par la suite), ainsi qu'un générateur g (*i.e.* $\mathcal{G} = \langle g \rangle$). On pourra penser à tout sous-groupe de $(\mathbb{Z}_p^\star, \times)$ d'ordre q pour $q \,|\, p-1$, ou à des courbes elliptiques, etc. Dans de tels groupes, on considère les problèmes suivants :

- le problème du **logarithme discret** (DL) : étant donné $y \in \mathcal{G}$, calculer $x \in \mathbb{Z}_q$ tel que $y = g^x = g \times \ldots \times g$ (x fois). On définit alors $\log_g y = x$, ainsi que le succès d'un algorithme $\mathcal{A}$ par
$$\mathsf{Succ}^{\mathsf{dl}}(\mathcal{A}) = \Pr_{x \in \mathbb{Z}_q}[\mathcal{A}(g^x) = x].$$
- le problème **Diffie–Hellman Calculatoire** (CDH) : étant donné deux éléments dans le groupe $\mathcal{G}$, $A = g^a$ et $B = g^b$, calculer $C = g^{ab}$. On définit alors $C = \mathsf{DH}(A, B)$ ainsi que le succès d'un algorithme $\mathcal{A}$ par
$$\mathsf{Succ}^{\mathsf{cdh}}(\mathcal{A}) = \Pr_{a,b \in \mathbb{Z}_q}[\mathcal{A}(g^a, g^b) = g^{ab}].$$
- le problème **Diffie–Hellman Décisionnel** (DDH) : étant donné trois éléments dans le groupe $\mathcal{G}$, $A = g^a$, $B = g^b$ et $C = g^c$, décider si $C = \mathsf{DH}(A, B)$, ce qui est équivalent à décider si $c = ab \bmod q$. On définit l'avantage d'un distingueur $\mathcal{D}$ par
$$\mathsf{Adv}^{\mathsf{ddh}}(\mathcal{D}) = |\Pr_{a,b,c \in \mathbb{Z}_q}[1 \leftarrow \mathcal{D}(g^a, g^b, g^c)] - \Pr_{a,b \in \mathbb{Z}_q}[1 \leftarrow \mathcal{D}(g^a, g^b, g^{ab})]\,|.$$

Ces problèmes sont classés du plus difficile au plus facile. En effet, la résolution du logarithme discret permet de résoudre les problèmes Diffie–Hellman. De même, il est plus facile de décider le DH que de le calculer.

De plus, ces problèmes sont aléatoirement auto-réductibles : toute instance peut se réduire à une instance aléatoire. Par exemple, si on veut calculer $x = \log_g y$, on peut choisir $a \in \mathbb{Z}_q$ aléatoire puis calculer $Y = yg^a$. Si l'on peut trouver $X = \log_g Y$, alors $x = X - a \bmod q$. Cette réduction convient quel que soit q. Une autre réduction est parfois utilisée : pour calculer $x = \log_g y$, on peut choisir $a \in \mathbb{Z}_q^\star$ aléatoire puis calculer $Y = y^a$. Si l'on peut trouver $X = \log_g Y$, alors $x = X/a \bmod q$. Cette réduction ne convient que si q est premier. Dans tous les cas, cette auto-réduction aléatoire signifie que toutes les instances sont aussi faciles/difficiles les unes que les autres : il n'y a que des instances moyennes. Ainsi, si on peut résoudre une fraction non-négligeable d'instances en temps polynomial, on peut résoudre toute instance en temps moyen polynomial.

Récemment, une nouvelle variante du problème Diffie–Hellman a été introduite, il s'agit du problème du *Gap Diffie-Hellman* (GDH) [14], qui consiste à résoudre le problème CDH avec un accès à un oracle DDH. Alors, on a
$$\mathsf{DL} \geq \mathsf{CDH} \geq \{\mathsf{DDH}, \mathsf{GDH}\},$$
où $A \geq B$ signifie que le problème A est au moins aussi difficile que le problème B. Cependant, en pratique, on ne sait résoudre aucun de ces problèmes autrement qu'en résolvant le problème du logarithme discret.

4.4.2 Difficulté du logarithme discret

L'algorithme le plus efficace pour résoudre le problème du logarithme discret dépend du groupe sous-jacent. En effet, pour les groupes dans lesquelles aucunes propriété algébrique

Soient p un nombre premier de la forme $\kappa q + 1$, où q est un grand nombre premier également (typiquement p est sur 512 bits et q sur 160 bits) et $g \in \mathbb{Z}_p^\star$ d'ordre q.

- Données Communes : p, q et g
- Clé privée d'Alice : $x \in \mathbb{Z}_q$
- Clé publique d'Alice : $y = g^x \bmod p$
- Chiffrement : soit m un message à chiffrer pour Alice. Ce message m est vu comme un élément de $\langle g \rangle$. Bob choisit un élément $k \in \mathbb{Z}_q$ puis calcule

$$r = g^k \bmod p \text{ et } s = y^k \times m \bmod p.$$

  Le chiffré de m est alors constitué de la paire (r, s).
- Déchiffrement : seule Alice est capable de retrouver m à partir du chiffré, grâce à sa connaissance de x. En effet,

$$y^k = g^{xk} = (g^k)^x = r^x \bmod p$$

  Ainsi, $m = s/r^x \bmod p$.

FIG. 4.5 – Chiffrement de El Gamal

spécifique ne peut être utilisée, seuls les algorithmes génériques sont effectifs [19, 16]. Leur complexité est en $\sqrt{q}$. Par exemple, sur les courbes elliptiques en général, seuls ces algorithmes peuvent être utilisés. Le dernier record en date a été établi en avril 2001 sur la courbe définie par l'équation $y^2 + xy = x^3 + x^2 + 1$ sur le corps fini à 2^{109} éléments.

En revanche, pour les sous-groupes de $\mathbb{Z}_p^\star$, des techniques plus efficaces peuvent être appliquées, en raison de la richesse de la structure. Le dernier record, établi en avril 2001 également, a calculé un logarithme discret modulo un entier premier de 120 chiffres décimaux.

4.5 Le cryptosystème de El Gamal

4.5.1 Description

En 1985, El Gamal a proposé un cryptosystème basé sur le problème du Logarithme Discret, ou plus précisément sur le problème Diffie-Hellman. Une description est donnée figure 4.5.

4.5.2 Résultats de sécurité

Théorème 4.7 *L'inversion (OW-CPA) du chiffrement de El Gamal est équivalente au problème Diffie-Hellman Calculatoire :* $\mathsf{Succ}^{\mathsf{ow-cpa}}(t) \leq \mathsf{Succ}^{\mathsf{cdh}}(t)$.

Preuve. Soit une instance aléatoire $(A = g^a, B = g^b)$ du problème Diffie-Hellman que l'on souhaite résoudre. Considérons l'attaquant $\mathcal{A}$ contre OW-CPA : nous dénommons le jeu réél que joue l'attaquant Game_0.

Game_0 **:** on exécute l'algorithme de génération de clés qui retourne $y = g^x$ pour un x aléatoire dans $\mathbb{Z}_q$. Puis l'attaquant reçoit un challenge $(r = g^k, s = my^k)$, pour un

message $m \stackrel{R}{\leftarrow} \mathcal{G}$ aléatoire. L'attaquant retourne $m = s/y^k$ avec probabilité ε. On note cet événement S_0, ainsi que S_i dans les jeux Game_i ci-dessous : $\Pr[S_0] = \varepsilon$.

Game_1 **:** on modifie un peu le jeu réel, notamment le choix de la clé publique. Au lieu de prendre $y = g^x$ comme clé publique, on utilise $y = A$, ce qui revient à prendre $x = a$. Les distributions de x et y sont identiques à celles du jeu précédent puisqu'il s'agit d'une instance (A, B) aléatoire : $\Pr[S_1] = \Pr[S_0]$.

Game_2 **:** on modifie désormais la construction du challenge. Au lieu de prendre $r = g^k$, on utilise $r = B$, ce qui revient à prendre $k = b$. La construction de s reste inchangée : $s = my^k$, pour un message $m \stackrel{R}{\leftarrow} \mathcal{G}$ aléatoire. La distribution de r est identique à celle du jeu précédent : $\Pr[S_2] = \Pr[S_1]$.

Game_3 **:** enfin, au lieu de définir $s = my^k$, pour un message $m \stackrel{R}{\leftarrow} \mathcal{G}$ aléatoire, on choisit $s \stackrel{R}{\leftarrow} \mathcal{G}$ aléatoire. La structure de groupe fait que la distribution de s est uniforme dans les deux cas : $\Pr[S_3] = \Pr[S_2]$.

On peut réécrire l'événement S_3 de la façon suivante :

$$\begin{aligned} \varepsilon &= \Pr[S_3] = \Pr[s \stackrel{R}{\leftarrow} \mathcal{G}, a, b \stackrel{R}{\leftarrow} \mathbb{Z}_q, y = g^a, r = g^b : \mathcal{A}(y, r, s) = s/y^b] \\ &= \Pr[A, B, s \stackrel{R}{\leftarrow} \mathcal{G}, y = A, r = B : \mathcal{A}(y, r, s) = s/\mathsf{DH}(A, B)]. \end{aligned}$$

La probabilité ε est donc bornée par la probabilité de résoudre le problème CDH, en le même temps que l'exécution de $\mathcal{A}$. □

Cet algorithme de chiffrement, contrairement aux schémas vus jusqu'à présent, a la particularité d'être probabiliste : il existe pleins de chiffrés possibles pour un même message clair, et ce en raison de l'aléa k. Il permet d'espérer la sécurité sémantique.

Théorème 4.8 *La sécurité sémantique (IND-CPA) du chiffrement de El Gamal est équivalente au problème Diffie-Hellman Décisionnel :* $\mathsf{Adv}^{\mathsf{ind-cpa}}(t) \leq 2 \times \mathsf{Adv}^{\mathsf{ddh}}(t)$.

Preuve. Comme ci-dessus, soit une instance aléatoire $(A = g^a, B = g^b)$ du problème Diffie-Hellman Décisionnel, avec C comme candidat. Considérons l'attaquant $\mathcal{A} = (A_1, A_2)$ contre $\mathsf{IND\text{-}CPA}$ en temps t : nous dénommons ce jeu réel Game_0.

Game_0 **:** on exécute l'algorithme de génération de clés qui retourne $y = g^x$ pour un x aléatoire dans $\mathbb{Z}_q$. Sur y, A_1 retourne deux messages (m_0, m_1). Sur le chiffré $\gamma = (r, s)$ de m_δ, A_2 retourne son choix δ'. Avec probabilité $(\varepsilon + 1)/2$, $\delta' = \delta$. On note cet événement S_0, ainsi que S_i dans les jeux Game_i ci-dessous : $\Pr[S_0] = (\varepsilon + 1)/2$.

Game_1 **:** comme ci-dessus, on modifie un peu le jeu réel, en utilisant $y = A$ puis $r = B$, ce qui revient à prendre $x = a$ et $k = b$. La construction de s reste inchangée : $s = m_\delta y^k$. Les distributions de x, y et r sont identiques, en raison de l'instance aléatoire (A, B) : $\Pr[S_1] = \Pr[S_0]$.

Game_2 **:** puis, au lieu de définir $s = m_\delta y^k$, on définit $s = m_\delta C$, pour $C = \mathsf{DH}(A, B)$. Alors, $\Pr[S_2] = \Pr[S_1]$.

Game_3 **:** maintenant, on remplace $C = \mathsf{DH}(A, B)$ par un candidat $C = g^c$ aléatoire. Puisque l'événement $\delta' = \delta$ est détectable, on peut définir le distingueur $\mathcal{D}$ qui exécute le même jeu que le Game_2, qui peut être effectivement le Game_2 ou le Game_3 selon que $C = \mathsf{DH}(A, B)$ ou non, ce que l'on ignore. Toujours est-il que le distingueur, à la fin du jeu retourne 0 si $\delta' \neq \delta$, et 1 si $\delta' = \delta$:

$$\Pr[1 \leftarrow \mathcal{D} \,|\, C \stackrel{R}{\leftarrow} \mathcal{G}] = \Pr[S_3] \text{ et } \Pr[1 \leftarrow \mathcal{D} \,|\, C = \mathsf{DH}(A, B)] = \Pr[S_2].$$

Ainsi,

$$|\Pr[S_3] - \Pr[S_2]| \leq \mathsf{Adv}^{\mathsf{ddh}}(\mathcal{D}) \leq \mathsf{Adv}^{\mathsf{ddh}}(t).$$

Enfin, il est aisé de remarquer que $\Pr[S_3] = 1/2$. Appliquons alors l'inégalité triangulaire :

$$\frac{\varepsilon}{2} = \frac{1+\varepsilon}{2} - \frac{1}{2} = |\Pr[S_3] - \Pr[S_0]| \leq \mathsf{Adv}^{\mathsf{ddh}}(t).$$

Donc l'avantage ε est bornée par deux fois l'avantage dans la décision du problème DDH, en le même temps que l'exécution de $\mathcal{A}$. □

Cependant, en raison de la propriété homomorphe, la non-malléabilité est inaccessible, ni même la moindre sécurité face à des attaques à chiffrés choisis adaptatives : si (r, s) est un chiffré de m, $(r, 2s)$ est un chiffré de $2m$.

4.6 Les attaques à chiffrés choisis adaptatives

4.6.1 Le modèle de l'oracle aléatoire

On a vu qu'il serait bien de résister aux attaques à chiffrés choisis. Néanmoins l'efficacité ne doit pas en pâtir. Ainsi, Bellare et Rogaway [5] ont introduit un modèle faisant l'hypothèse que certaines fonctions sont parfaitement aléatoires. Cela revient aussi à dire que l'attaque est indépendante de l'implémentation effective de la fonction. Plus formellement, dans toutes les probabilités, la distribution aléatoire de certaines fonctions est ajoutée à l'espace de probabilités.

4.6.2 Première construction générique

Ils ont d'ailleurs proposé une construction générique permettant de construire un cryptosystème IND-CCA2 à partir de toute permutation à sens-unique à trappe.

Cette construction fait appel à deux oracles aléatoires G et H, à valeurs dans $\{0,1\}^n$ et $\{0,1\}^{k_1}$ respectivement. L'algorithme de génération des clés définit une permutation f de l'espace E comme clé publique, et son inverse g comme clé privée (possible grâce à la trappe). Pour chiffrer un message $m \in \{0,1\}^n$, on choisit $r \xleftarrow{R} E$, puis on calcule

$$\mathcal{E}(m; r) = f(r) \,\|\, m \oplus G(r) \,\|\, H(m, r).$$

Le déchiffrement d'un chiffré $C = a \,\|\, b \,\|\, c$ s'effectue en deux étapes : tout d'abord, on retrouve $r = g(a)$, grâce à la trappe de f, puis $m = b \oplus G(r)$; ensuite, avant de retourner le message m, on vérifie la consistance du chiffré, à savoir si $c = H(m, r)$.

Au sujet de cette construction, on peut montrer le résultat suivant :

Théorème 4.9 *Considérons un adversaire $\mathcal{A}$ contre cette construction, selon une attaque à chiffrés choisis adaptative. Supposons qu'après q_D questions à l'oracle de déchiffrement et q_G, q_H questions aux oracles G et H, $\mathcal{A}$ peut avoir un avantage ε en temps t, alors on peut inverser f avec succès $\varepsilon/2 - q_D/2^{k_1}$, en temps $t + (q_G + q_H)T_f$, où T_f désigne le temps d'une évaluation de f.*

Avant de prouver ce théorème, établissons le lemme suivant :

Lemme 4.3 *Soient E, F et G des événements dans un espace de probabilités, alors*

$$\Pr[\mathsf{E} \wedge \neg\mathsf{G}] = \Pr[\mathsf{F} \wedge \neg\mathsf{G}] \implies |\Pr[\mathsf{E}] - \Pr[\mathsf{F}]| \leq \Pr[\mathsf{G}].$$

Preuve. La différence $|\Pr[\mathsf{E}] - \Pr[\mathsf{F}]|$ est égale à

$$|\Pr[\mathsf{E} \wedge \neg\mathsf{G}] + \Pr[\mathsf{E} \wedge \mathsf{G}] - \Pr[\mathsf{F} \wedge \neg\mathsf{G}] - \Pr[\mathsf{F} \wedge \mathsf{G}]| = |\Pr[\mathsf{E} \wedge \mathsf{G}] - \Pr[\mathsf{F} \wedge \mathsf{G}]|$$
$$= |\Pr[\mathsf{E} \mid \mathsf{G}] \cdot \Pr[\mathsf{G}] - \Pr[\mathsf{F} \mid \mathsf{G}] \cdot \Pr[\mathsf{G}]| \leq |\Pr[\mathsf{E} \mid \mathsf{G}] - \Pr[\mathsf{F} \mid \mathsf{G}]| \cdot \Pr[\mathsf{G}] \leq \Pr[\mathsf{G}].$$

□

Preuve. Considérons l'attaquant $\mathcal{A} = (A_1, A_2)$ contre ce schéma. Dans les deux étapes, A_1 et A_2 ont accès à l'oracle de déchiffrement.

Game_0 **:** on exécute l'algorithme de génération de clés qui retourne une permutation f et son inverse g. On génère également $x \xleftarrow{R} E$ et $y = f(x)$. Après avoir vu la clé publique (la description de la fonction f), A_1 retourne deux messages m_0 et m_1. Après avoir reçu le chiffré $C = a \,\|\, b \,\|\, c$ du message m_δ, A_2 retourne un bit δ'. On note r l'unique élément tel que $C = \mathcal{E}(m_\delta, r)$. Avec probabilité $(\varepsilon + 1)/2$, $\delta' = \delta$. On note cet événement S_0, ainsi que S_i dans les jeux Game_i ci-dessous : $\Pr[S_0] = (1 + \varepsilon)/2$.

Pour la suite, on pourra supposer que toute question $H(\star, \rho)$ est précédée de la question $G(\rho)$.

Game_1 **:** dans un premier temps, on remplace les oracles G et H par des simulations classiques : pour toute nouvelle question à l'un des ces oracles, on répond par une chaîne aléatoire dans l'espace correspondant, puis on stocke les questions-réponses dans les listes Liste_G et Liste_H respectivement. Il s'agit de simulations parfaites, $\Pr[S_1] = \Pr[S_0]$.

Game_2 **:** dans ce jeu, on simule l'oracle de déchiffrement. À la question $C' = a' \,\|\, b' \,\|\, c'$, pour $a' = f(r')$, si r' n'est pas dans Liste_G, on rejette le chiffré ; si de même $(b' \oplus G(r'), r')$ n'est pas non plus dans Liste_H, on rejette le chiffré ; dans les autres cas, on continue à utiliser l'oracle de déchiffrement.

Avec l'hypothèse ci-dessus, on ne peut refuser un chiffré valide que si $(b' \oplus G(r'), r')$ n'a pas été demandé à H. Mais alors, $H(b' \oplus G(r'), r')$ retourne un élément parfaitement aléatoire, qui est égal à c' avec probabilité $1/2^{k_1}$. Ainsi, $|\Pr[S_2] - \Pr[S_1]| \leq q_D/2^{k_1}$.

Game_3 **:** on poursuit la simulation de l'oracle de déchiffrement, sur $C' = a' \,\|\, b' \,\|\, c'$, pour $a' = f(r')$. On sait que $r' \in \mathsf{Liste}_G$ et $(b' \oplus G(r'), r') \in \mathsf{Liste}_H$. On peut alors trouver ce r' (en testant si $f(r') = a'$ sur toutes les questions à G, grâce à la propriété de permutation de f), puis déchiffrer correctement : $\Pr[S_3] = \Pr[S_2]$.

Game_4 **:** dans ce jeu, on définit $a = y = f(x)$, $b = m_\delta \oplus g^+$ et $c = h^+$, où x, g^+ et h^+ sont aléatoires. De plus, à la question $G(x)$, on répond g^+, et à la question $H(m_\delta, x)$ on répond h^+. Il s'agit simplement de spécifier certaines valeurs de G et H, par des valeurs aléatoires, on ne modifie donc pas les distributions : $\Pr[S_4] = \Pr[S_3]$.

Game_5 **:** maintenant, on supprime les modifications locales de G et H. Les réponses aux question $G(x)$ et $H(m_\delta, x)$ sont indépendantes de x et m_δ. La seule différence apparaît si l'événement « x a été demandé », nommé G, a lieu : $|\Pr[S_5] - \Pr[S_4]| \leq \Pr[\mathsf{G}]$. Cependant, dans ce dernier jeu, δ est indépendant de la vue de l'attaquant, ainsi $\Pr[S_5] = 1/2$.

À nouveau, l'inégalité triangulaire nous donne

$$\frac{\varepsilon}{2} = \frac{1 + \varepsilon}{2} - \frac{1}{2} = |\Pr[S_0] - \Pr[S_5]| \leq \frac{q_D}{2^{k_1}} + \Pr[\mathsf{G}].$$

Cependant, l'événement G permet d'inverser $f(x)$ en testant toutes les questions posées à G et H, d'où le résultat. □

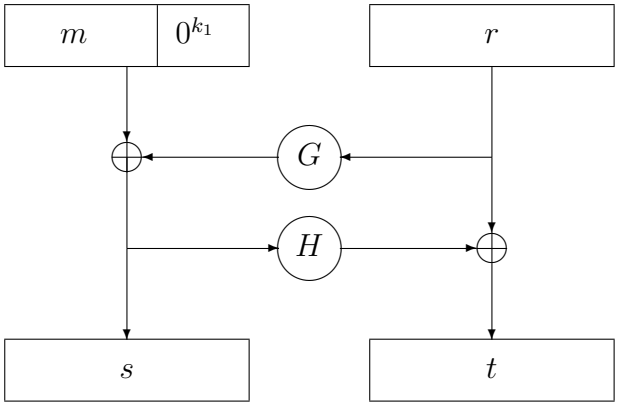


FIG. 4.6 – Optimal Asymmetric Encryption Padding

4.6.3 Constructions plus génériques

Bellare et Rogaway ont ensuite présenté une autre construction générique plus efficace, et surtout produisant un chiffré plus court, appelée OAEP (Optimal Asymmetric Encryption Padding) [6, 20, 10]. Il s'agit d'appliquer la transformation présentée figure 4.6 au message m à chiffrer, avec un aléa r, puis d'appliquer la permutation f au résultat $s\|t$. Le déchiffrement s'effectue en deux temps : retrouver $s\|t$ grâce à l'inverse de f, puis retrouver le message m, qui est retourné si la redondance de k_1 bits à 0 est satisfaite. Malheureusement, cette construction se limite encore une fois aux permutations, et la seule application est essentiellement RSA.

De nouvelles constructions ont plus récemment été proposées, pour s'appliquer à des fonctions à sens-unique à trappe, avec une certaine injectivité. La plus efficace est REACT (Rapid Enhanced-security Asymmetric Cryptosystem Transform) [15].

Cette construction (présentée figure 4.7) fait appel à deux oracles aléatoires G et H, à valeurs dans $\{0,1\}^n$ et $\{0,1\}^{k_1}$ respectivement. L'algorithme de génération des clés définit une fonction probabiliste injective f de X dans Y, comme clé publique, et son inverse g comme clé privée (possible grâce à la trappe). Une telle fonction probabiliste f fait intervenir un aléa α pour calculer une image y de son entrée x. La valeur de cet aléa modifiera éventuellement la valeur de sortie. L'injectivité d'une telle fonction signifie que pour tout $y \in Y$, il existe au plus une entrée x qui puisse conduire à y (avec certaines valeurs d'aléa). Pour chiffrer un message $m \in \{0,1\}^n$, on choisit $r \xleftarrow{R} X$, puis on calcule

$$a = f(r), b = m \oplus G(r), c = H(a, b, m, r) \text{ puis } \mathcal{E}(m; r) = a \,\|\, b \,\|\, c.$$

Le déchiffrement d'un chiffré $C = a \,\|\, b \,\|\, c$ s'effectue en deux étapes : tout d'abord, on retrouve $r = g(a)$, grâce à la trappe de f, puis $m = b \oplus G(r)$; ensuite, avant de retourner le message m, on vérifie la consistance du chiffré, à savoir si $c = H(a, b, m, r)$. Au sujet de cette construction, on peut montrer le résultat suivant :

Théorème 4.10 *Considérons un adversaire $\mathcal{A}$ contre cette construction, selon une attaque à chiffrés choisis adaptative. Supposons qu'après q_D questions à l'oracle de déchiffrement et q_G, q_H questions aux oracles G et H, $\mathcal{A}$ peut avoir un avantage ε en temps t, alors on peut inverser f avec succès $\varepsilon/2 - q_D/2^{k_1}$, en temps $t + (q_G + q_H)T_f$, avec $q_G + q_H$ tests $f^{-1}(y) \stackrel{?}{=} x$, où T_f désigne le temps nécessaire pour un tel test.*

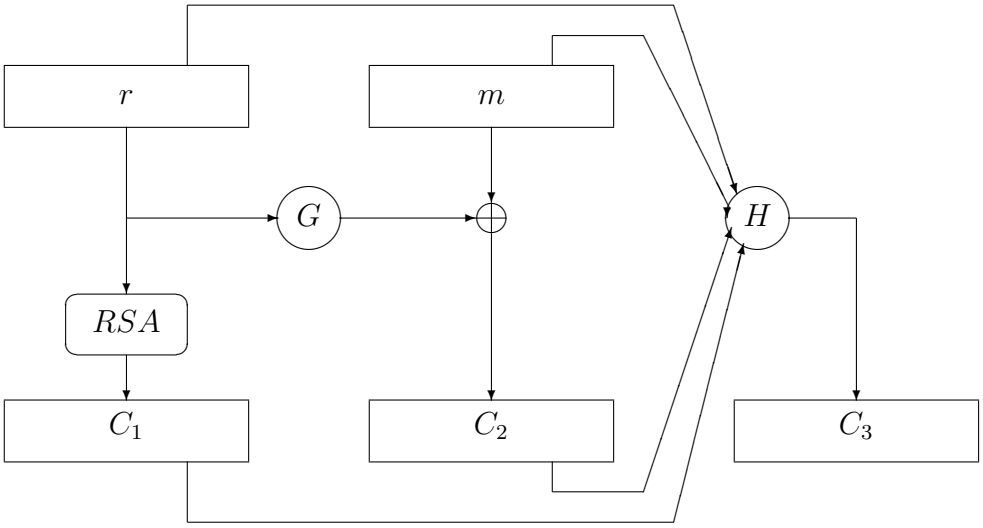


FIG. 4.7 – Rapid Enhanced-security Asymmetric Cryptosystem Transform

Preuve. La preuve est très similaire à la preuve précédente. Considérons l'attaquant $\mathcal{A} = (A_1, A_2)$ contre ce schéma. Dans les deux étapes, A_1 et A_2 ont accès à l'oracle de déchiffrement.

Game_0 **:** on exécute l'algorithme de génération de clés qui retourne une permutation f et son inverse g. On génère également $x \xleftarrow{R} X$ et $y = f(x)$. Après avoir vu la clé publique (la description de la fonction f), A_1 retourne deux messages m_0 et m_1. Après avoir reçu le chiffré $C = a \,\|\, b \,\|\, c$ du message m_δ, A_2 retourne un bit δ'. On note r l'unique élément tel que $C = \mathcal{E}(m_\delta, r)$. Avec probabilité $(\varepsilon + 1)/2$, $\delta' = \delta$. On note cet événement S_0, ainsi que S_i dans les jeux Game_i ci-dessous : $\Pr[S_0] = (1 + \varepsilon)/2$.

Pour la suite, on pourra supposer que toute question $H(\star, \star, \star, \rho)$ est précédée de la question $G(\rho)$.

Game_1 **:** dans un premier temps, on remplace les oracles G et H par les simulations classiques parfaites : $\Pr[S_1] = \Pr[S_0]$.

Game_2 **:** dans ce jeu, on simule l'oracle de déchiffrement. À la question $C' = a' \,\|\, b' \,\|\, c'$, pour $a' = f(r')$, si r' n'est pas dans Liste_G, ce que l'on détecte par un test $f^{-1}(a') \stackrel{?}{=} r'$, on rejette le chiffré ; si de même $(a', b', b' \oplus G(r'), r')$ n'est pas non plus dans Liste_H, ce que l'on détecte par le même type de test, on rejette le chiffré ; dans les autres cas, on continue à utiliser l'oracle de déchiffrement.

Avec l'hypothèse ci-dessus, on ne peut refuser un chiffré valide que si le quadruplet $(a', b', b' \oplus G(r'), r')$ n'a pas été demandé à H. Mais alors, $H(a', b', b' \oplus G(r'), r')$ retourne une valeur parfaitement aléatoire qui est égale à c' avec probabilité $1/2^{k_1}$. Ainsi, $|\Pr[S_2] - \Pr[S_1]| \leq q_D/2^{k_1}$.

Game_3 **:** on poursuit la simulation de l'oracle de déchiffrement, sur $C' = a' \,\|\, b' \,\|\, c'$, pour $a' = f(r')$. On sait que $r' \in \mathsf{Liste}_G$ et $(a', b', b' \oplus G(r'), r') \in \mathsf{Liste}_H$. On peut alors trouver ce r' (en testant si $f^{-1}(a') = r'$ sur toutes les questions à G, grâce à l'oracle de test), puis déchiffrer correctement : $\Pr[S_3] = \Pr[S_2]$.

Game_4 **:** dans ce jeu, on définit $a = y = f(x)$, $b = m_\delta \oplus g^+$ et $c = h^+$, où x, g^+ et h^+ sont aléatoires. De plus, à la question $G(x)$, on répond g^+, et à la question $H(a, b, m_\delta, x)$

on répond h^+. Il s'agit simplement de spécifier certaines valeurs de G et H, par des valeurs aléatoires, on ne modifie donc pas les distributions : $\Pr[S_4] = \Pr[S_3]$.

Game_5 **:** maintenant, on supprime les modifications locales de G et H. Les réponses aux question $G(x)$ et $H(a, b, m_\delta, x)$ sont indépendantes de x et m_δ. La seule différence apparaît si l'événement « x a été demandé », nommé G, a lieu. Par conséquent, $|\Pr[S_5] - \Pr[S_4]| \leq \Pr[\mathsf{G}]$. Cependant, dans ce dernier jeu, δ est indépendant de la vue de l'attaquant, ainsi $\Pr[S_5] = 1/2$.

À nouveau, l'inégalité triangulaire nous donne

$$\frac{\varepsilon}{2} = \frac{1+\varepsilon}{2} - \frac{1}{2} = |\Pr[S_0] - \Pr[S_5]| \leq \frac{q_D}{2^{k_1}} + \Pr[\mathsf{G}].$$

Cependant, l'événement G permet d'inverser $f(x)$ en testant toutes les questions posées à G et H, d'où le résultat. □

Cette construction a l'avantage d'être beaucoup plus générale, elle s'applique notamment à la fonction El Gamal : $f_Y : \mathcal{G} \to \mathcal{G} \times \mathcal{G}$ qui sur l'entrée $m \in \mathcal{G}$, avec un aléa $x \in \mathbb{Z}_q$, produit le couple $(r, s) = (g^x, m \times Y^x)$. L'inversion de cette fonction est possible pour qui connaît $X = \log_g Y$: $f^{-1}(r, s) = s/r^X$. La difficulté de l'inversion de cette fonction probabiliste f_Y repose bien sûr sur le problème CDH. La difficulté du test $f_Y^{-1}(r, s) \stackrel{?}{=} = m$ repose en revanche sur le problème DDH. Ainsi, inverser la fonction avec un accès à un oracle de test consiste à casser le problème GDH.

4.6.4 Le chiffrement de Cramer-Shoup

En 1998, Cramer et Shoup [8] ont proposé le premier schéma de chiffrement asymétrique résistant aux attaques à chiffrés choisis adaptatives sans faire intervenir le modèle de l'oracle aléatoire. Il s'agit d'une variante du chiffrement El Gamal. On se place alors dans un groupe $\mathcal{G} = \langle g \rangle$ d'ordre premier q. On a également besoin d'une fonction de hachage H supposée résistante aux collisions (même modulo q). La clé privée d'Alice consiste en quatre éléments $\omega, x, y, z \in \mathbb{Z}_q$. Quant à sa clé publique, elle est construite de la façon suivante :

$$g_1 = g, g_2 = g_1^\omega, c = g_1^x, d = g_1^y \text{ et } h = g_1^z.$$

Pour chiffrer un message m, vu comme un élément de $\mathcal{G}$. Bob choisit un élément $r \in \mathbb{Z}_q$ puis calcule

$$u_1 = g_1^r, u_2 = g_2^r, e = h^r m, \alpha = H(u_1, u_2, e) \text{ et } v = c^r d^{\alpha r}.$$

Le chiffré est constitué du quadruplet (u_1, u_2, e, v).

Pour déchiffrer un tel quadruplet, Alice commence par en vérifier la validité : $u_2 = u_1^\omega$ puis $v = u_1^{x+\alpha y}$, après avoir calculé α. Si les deux tests sont satisfaits, le message clair est $m = e/u_1^z$.

Théorème 4.11 *La sécurité sémantique contre des attaques à chiffrés choisis (IND-CCA2) du chiffrement Cramer-Shoup est équivalente au problème Diffie-Hellman Décisionnel, sous réserve de la résistance aux collisions de la fonction H.*

Le niveau de sécurité de ce schéma est remarquable. Reste à savoir si le sur-coût calculatoire est acceptable.

Bibliographie

[1] O. Baudron, D. Pointcheval, & J. Stern. Extended Notions of Security for Multicast Public Key Cryptosystems. In *Proc. of the 27th ICALP*, LNCS 1853, pages 499–511. Springer-Verlag, Berlin, 2000.

[2] M. Bellare, A. Boldyreva, & S. Micali. Public-key Encryption in a Multi-User Setting : Security Proofs and Improvements. In *Proc. of EUROCRYPT '2000*, LNCS 1807, pages 259–274. Springer-Verlag, Berlin, 2000.

[3] M. Bellare, A. Desai, D. Pointcheval, & P. Rogaway. Relations among Notions of Security for Public-Key Encryption Schemes. In *Proc. of CRYPTO '98*, LNCS 1462, pages 26–45. Springer-Verlag, Berlin, 1998.

[4] M. Bellare, C. Namprempre, D. Pointcheval, and M. Semanko. The Power of RSA Inversion Oracles and the Security of Chaum's RSA Blind Signature Scheme. In *Proc. of Financial Cryptography '2001*, LNCS. Springer-Verlag, Berlin, 2001.

[5] M. Bellare & P. Rogaway. Random Oracles Are Practical : a Paradigm for Designing Efficient Protocols. In *Proc. of the 1st CCS*, pages 62–73. ACM Press, New York, 1993.

[6] M. Bellare and P. Rogaway. Optimal Asymmetric Encryption – How to Encrypt with RSA. In *Proc. of EUROCRYPT '94*, LNCS 950, pages 92–111. Springer-Verlag, Berlin, 1995.

[7] D. Bleichenbacher. A Chosen Ciphertext Attack against Protocols based on the RSA Encryption Standard PKCS #1. In *Proc. of CRYPTO '98*, LNCS 1462, pages 1–12. Springer-Verlag, Berlin, 1998.

[8] R. Cramer and V. Shoup. A Practical Public Key Cryptosystem Provably Secure against Adaptive Chosen Ciphertext Attack. In *Proc. CRYPTO '98*, LNCS 1462, pages 13–25. Springer-Verlag, Berlin, 1998.

[9] D. Dolev, C. Dwork, and M. Naor. Non-Malleable Cryptography. *SIAM Journal on Computing*, 30(2) :391–437, 2000.

[10] E. Fujisaki, T. Okamoto, D. Pointcheval, and J. Stern. RSA–OAEP is Secure under the RSA Assumption. In *Proc. CRYPTO '2001*, LNCS 2139, pages 260–274. Springer-Verlag, Berlin, 2001.

[11] S. Goldwasser and S. Micali. Probabilistic Encryption. *Journal of Computer and System Sciences*, 28 :270–299, 1984.

[12] J. Håstad. Solving Simultaneous Modular Equations of Low Degree. *SIAM Journal of Computing*, 17 :336–341, 1988.

[13] M. Naor and M. Yung. Public-Key Cryptosystems Provably Secure against Chosen Ciphertext Attacks. In *Proc. of the 22nd ACM STOC*, pages 427–437. ACM Press, New York, 1990.

[14] T. Okamoto and D. Pointcheval. The Gap-Problems : a New Class of Problems for the Security of Cryptographic Schemes. In *Proc. of PKC '2001*, LNCS 1992, pages 104–118. Springer-Verlag, Berlin, 2001.

[15] T. Okamoto and D. Pointcheval. REACT : Rapid Enhanced-security Asymmetric Cryptosystem Transform. In *Proc. of RSA '2001*, LNCS 2020, pages 159–175. Springer-Verlag, Berlin, 2001.

[16] J. M. Pollard. Monte Carlo Methods for Index Computation (mod p). *Mathematics of Computation*, 32(143) :918–924, juillet 1978.

[17] C. Rackoff and D. R. Simon. Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack. In *Proc. of CRYPTO '91*, LNCS 576, pages 433–444. Springer-Verlag, Berlin, 1992.

[18] R. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public Key Cryptosystems. *Communications of the ACM*, 21(2) :120–126, février 1978.

[19] D. Shanks. Class Number, a Theory of Factorization, and Genera. In *Proc. of the Symposium on Pure Mathematics*, volume 20, pages 415–440. AMS, 1971.

[20] V. Shoup. OAEP Reconsidered. In *Proc. of CRYPTO '2001*, LNCS 2139, pages 239–259. Springer-Verlag, Berlin, 2001.

Chapitre 5 (4h)

Zero-knowledge et identification

— par Jacques Stern

Sommaire

5.1 Motivation et exemple 93
5.1.1 Le scénario du zero-knowledge 93
5.1.2 Un exemple : le coloriage d'un graphe 94
5.2 Approche formelle du ZK 96
5.2.1 Définitions 96
5.2.2 Preuves ZK de problèmes NP 97
5.3 Preuves ZK d'identité 98
5.3.1 Racines carrées modulo un entier RSA 98
5.3.2 Le schéma d'identification de Fiat-Shamir 99
5.3.3 Preuve du schéma de Fiat-Shamir 99
5.3.4 Le contexte cryptographique 101
5.3.5 Le Fiat-Shamir à plusieurs secrets 102
5.3.6 Le schéma de Guillou-Quisquater 103

5.1 Motivation et exemple

5.1.1 Le scénario du zero-knowledge

La notion de prédicat polynomial a été introduite dans la section 1.5.1. Soit $R(x, y)$ un tel prédicat. On s'intéresse ici à une situation dans laquelle deux entités ont en commun une donnée x : la première, appelée le *prouveur* dispose de plus d'une solution y au problème NP associé, c'est-à-dire d'une valeur y telle que $R(x, y)$ soit vrai. Une telle valeur prend le nom de *témoin*. Typiquement, il s'agit d'une situation cryptographique, où le témoin y est un secret propre au prouveur, relié à la donnée publique x par une relation simple. Le prouveur interagit avec l'autre entité, appelée *vérifieur*, et doit le convaincre de ce qu'il possède un témoin y, sans toutefois révéler la moindre information sur y.

Les tailles de x et y sont contrôlées par un paramètre de sécurité k. Le prouveur et le vérifieur sont modélisés par des MTPPI (machines de Turing probabilistes polynomiales interactives) dont la complexité s'évalue en fonction de k et ceci rend bien compte d'un processus de questions/réponses. A l'issue de l'interaction, le vérifieur prend une décision d'acceptation ou de rejet :

– face à un prouveur qui a bien accès à un témoin, le vérifieur accepte ;
– dans le cas où il accepte, le vérifieur est convaincu de ce qu'il est en présence d'un prouveur en possession d'un témoin.

La "conviction" est de nature probabiliste : avec une très forte probabilité, le vérifieur rejette tout faussaire. Une définition plus formelle sera donnée plus loin.

La préservation du secret de y passe par une simulation conformément aux idées présentées dans la section 1.5.2. La vue d'un adversaire est constituée de l'ensemble des communications échangées entre le prouveur et le vérifieur et on requiert que cette vue soit simulable. Là encore, une définition précise sera donnée plus loin. On note toutefois qu'il n'y a pas de raison d'être optimiste et de se restreindre au cas d'un vérifieur "honnête". Le concept de zero-knowledge (ZK), que l'on peut traduire en français par *"apport nul de connaissance"*, exige que la simulation reste possible si le vérifieur est remplacé par n'importe quelle MTPPI.

5.1.2 Un exemple : le coloriage d'un graphe

On rappelle qu'un graphe non orienté est une relation binaire symétrique sur un domaine fini. Les éléments du domaine sont les *sommets* et les paires qui satisfont la relation, appelées les *arêtes*. On se donne un ensemble fini à trois éléments, par exemple les trois "couleurs" rouge (R), bleu (B), vert (V). Un *coloriage* est une application c de l'ensemble $\mathcal{V}$ des sommets dans $\{R, B, V\}$ telle que pour toute arête $\{u, v\}$, on ait $c(u) \neq c(v)$. En d'autres termes, deux sommets voisins reçoivent des couleurs distinctes. On voit bien le rapport avec le coloriage des cartes géographiques. le problème du coloriage d'un graphe à l'aide de trois couleurs est NP-complet (voir [3]).

On considère maintenant un prouveur qui a accès à un coloriage d'un graphe G. On va montrer comment il peut convaincre un vérifieur qui a accès au graphe seul, de façon zero-knowledge. On aura besoin de réaliser une fonctionnalité, appelée *engagement*, permettant à une entité de fixer de manière unique une valeur secrète s destinée à être révélée ultérieurement. C'est, en somme l'analogue, symbolique, d'un dépôt sous enveloppe dans le coffre d'un notaire. On requiert que l'engagement ne donne pas d'information sur s. Dans ce paragraphe un peu informel, on va réaliser cette fonctionnalité en appliquant une fonction de hachage H à $(s||r)$, où r est un aléa et $||$ désigne la concaténation. Il est à noter que, si s varie dans un ensemble petit, on ne peut se contenter de produire $H(s)$, puisqu'une recherche exhaustive permettrait de recouvrer s.

Soit donc c un coloriage de G accessible au prouveur. On note que, si π est une permutation des trois couleurs $\{R, B, V\}$, $\pi \circ c$ est également un coloriage de G. On procède aux échanges suivants :

1. Le prouveur choisit une permutation π des trois couleurs $\{R, B, V\}$ et transmet au vérifieur une liste d'engagements $e_u = H(\pi(c(u))||r_u)$, $u \in \mathcal{V}$, où les aléas r_u sont tirés aléatoirement et de manière indépendante.
2. Le vérifieur choisit aléatoirement une arête $\{u, v\}$ de G et la transmet au prouveur.
3. Le prouveur révèle au vérifieur $\pi(c(u))$, $\pi(c(v))$, r_u et r_v.
4. Le vérifieur teste les égalités $H(\pi(c(u))||r_u) = e_u$, $H(\pi(c(v))||r_v) = e_v$ et l'inégalité $\pi(c(u)) \neq \pi(c(v))$.

Les étapes ci-dessus 1 à 4 sont répétées un nombre suffisant t de fois. A l'issue de ces répétitions, le vérifieur accepte si tous les tests réalisés aux étapes 4 ont reçu une réponse

positive ; dans le cas contraire, le vérifieur rejette la preuve. Les étapes 1 à 4 sont commodément résumées dans la figure 5.1.

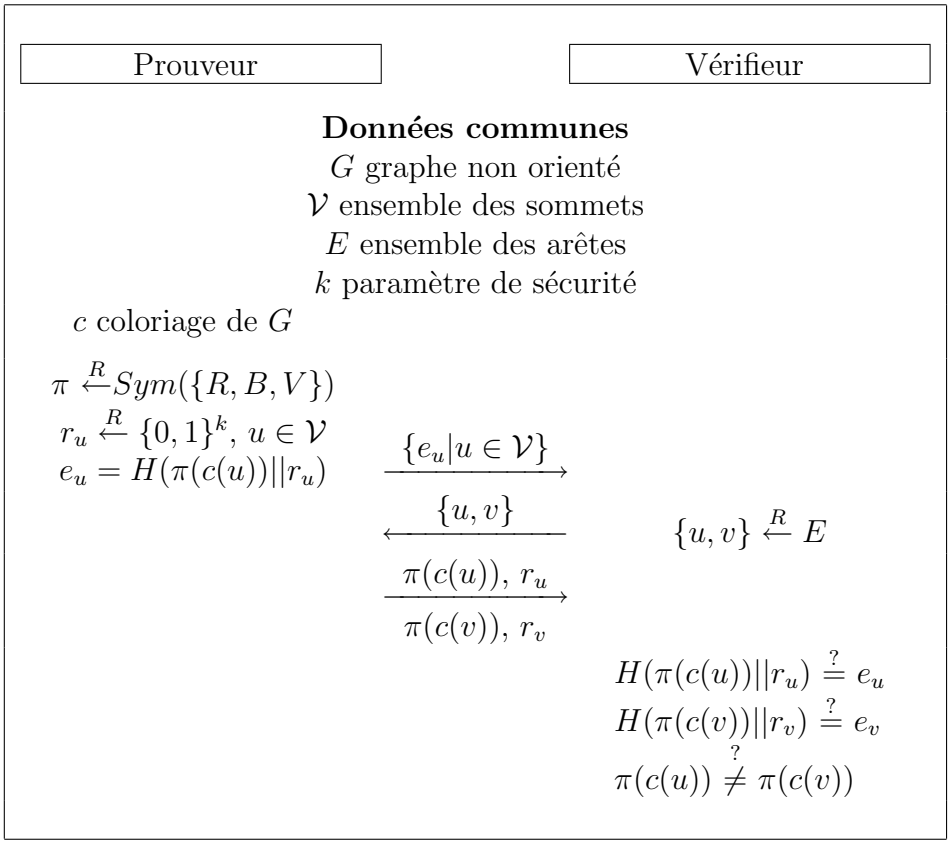


FIG. 5.1 – Preuve ZK de coloriage d'un graphe

Il est clair qu'un prouveur qui dispose d'un coloriage de G et qui suit exactement le protocole sait répondre correctement aux questions du vérifieur. On traite les autres propriétés du zero-knowledge de manière informelle. On y reviendra dans la suite. Un faussaire qui ne dispose pas d'un coloriage de G peut parfaitement passer l'étape 1 et l'étape 2. A l'étape 3, il ne peut répondre correctement à toutes les questions du vérifieur : en effet, il ne peut revenir sur les engagements qu'il a pris, puisque cela correspondrait à produire des collisions pour une fonction de hachage. L'ensemble des réponses à toutes les questions possibles induit donc un coloriage incorrect de G. La probabilité de passer les étapes 3 et 4 est donc au plus $1 - 1/|E|$. Au bout de t répétitions, on obtient une probabilité au plus $(1 - 1/|E|)^t$ qui, si t est assez grand, est de l'ordre de $\exp(-t/|E|)$, et peut donc être rendue exponentiellement petite, par exemple en prenant $t = |\mathcal{V}|^3$.

Reste la propriété de simulation. Elle est fondée sur l'anticipation de la question $\{u, v\}$ posée par le vérifieur à l'étape 2. Le prouveur prépare ses engagements de manière à pouvoir répondre à cette question précisément. C'est facile : il suffit de colorier u et v par des couleurs distinctes, de compléter le coloriage n'importe comment, sans respecter la règle de voisinage, et de calculer les engagements. On fait ensuite "tourner" le vérifieur (honnête ou non). Si la question qu'il pose a été correctement anticipée, on passe aux étapes 3 et 4. Sinon, on réitère la simulation en réalisant une nouvelle anticipation. Dans la mesure où les engagements ne donnent pas d'information sur les valeurs mises en gage, le vérifieur retourne la question attendue avec probabilité $\frac{1}{|E|}$. La simulation de quatre étapes prend donc en moyenne $|E|$

essais, après quoi on passe à l'itération suivante.

Les arguments ci-dessus, outre qu'ils ne s'appuient pas sur des définitions précises, ne traitent pas les probabilités de manière rigoureuse. On bute de plus sur l'utilisation de la fonction de hachage H, qu'il faudrait modéliser par un oracle aléatoire, au sens de la section 1.5.1. On va maintenant proposer une approche plus formelle.

5.2 Approche formelle du ZK

5.2.1 Définitions

Soit $R(x, y)$ un prédicat polynomial (en fonction d'un paramètre de sécurité k). On considère comme dans la section 5.1.1, deux MTPPI appelées respectivement prouveur et vérifieur, telles que :

– Le prouveur prend en entrée une donnée x et un témoin y.

– Le vérifieur prend en entrée la valeur de x seule.

Le couple formé des deux machines constitue une preuve ZK pour le problème NP associé à R si trois propriétés, nommées *complétude*, *correction* et *zero-knowledge*, sont satisfaites. Notre exposition suit l'article [1] (voir aussi [6]).

Complétude. Cette propriété exprime que l'exécution de l'interaction entre un prouveur et un vérifieur conduit à l'acceptation dans tous les cas. On affaiblit parfois cette définition en autorisant des exceptions de probabilité négligeable.

Correction. Cette propriété formalise la notion de "connaissance" d'un témoin. Pour l'énoncer, on considère une interaction entre une machine $\tilde{P}$, recevant en entrée x seul, et le vérifieur. On fait l'hypothèse que cette interaction conduit à l'acceptation avec probabilité non négligeable. On cherche alors à conclure qu'il existe une MTPP qui calcule un témoin avec probabilité non négligeable, en utilisant $\tilde{P}$ comme oracle. On appelle une telle machine un *extracteur*. Il existe de nombreuses variantes de cette définition dans la littérature. Celle qu'on présente ici a en vue la mise en œuvre de preuves par réduction au sens de la section 1.5.2. Il convient de préciser un peu ce qu'on entend par l'utilisation d'une MTPPI comme oracle : au lieu de donner accès au résultat du calcul, comme pour le cas des oracles ordinaires, on permet, en fournissant à l'oracle les données et les aléas dont il a besoin, d'obtenir successivement ce que chaque étape d'interaction produit.

Zero-knowledge. On considère ici l'interaction entre le prouveur, recevant en entrée x et y et une MTPPI quelconque $\tilde{V}$. La suite des communications échangées est une variable aléatoire, dépendant des aléas des deux machines et qu'on appelle *vue*. On requiert que cette vue soit simulable, c'est à dire qu'il existe une MTPP, prenant en entrée x seul, qui produise une vue essentiellement identique. Conformément aux définitions introduites dans la section 1.5.2, il y a en fait trois formes distinctes de ZK, appelées respectivement ZK parfait, ZK statistique, ZK algorithmique, suivant que l'expression "essentiellement identique" se réfère à une simulation parfaite, statistique ou algorithmique.

Une dernière remarque sur le sens du mot négligeable : dans l'approche traditionnelle de la complexité, est négligeable une quantité dépendant du paramètre de sécurité qui tend vers zéro plus vite que l'inverse de tout polynôme. Une tendance plus récente en cryptographie s'applique à donner des majorations optimales, plus facilement interprétables en pratique.

5.2.2 Preuves ZK de problèmes NP

On appelle *fonction à sens unique* une fonction f, définie sur les suites de bits de longueur quelconque, telle que :

i) Pour une donnée x de $\{0,1\}^k$, $f(x)$ est calculable en temps polynomial en fonction du paramètre de sécurité k.

ii) Aucune machine de Turing polynomiale probabiliste n'inverse f avec une probabilité non négligeable.

En d'autres termes, pour tout algorithme polynomial probabiliste $\mathcal{A}$, la probabilité

$$\Pr\left[f(\mathcal{A}(f(x))) = f(x) | x \stackrel{R}{\leftarrow} \{0,1\}^k\right]$$

décroît plus vite que l'inverse de tout polynôme :

$$(\forall c)(\exists \ell)(\forall k > \ell) \quad \Pr\left[f(\mathcal{A}(f(x))) = f(x) | x \stackrel{R}{\leftarrow} \{0,1\}^k\right] \leq \frac{1}{k^c}$$

L'existence d'une fonction à sens unique est une conjecture qui implique $P \neq NP$. Elle est indispensable à la cryptographie, qui se doit de considérer l'application $p, q \longrightarrow n = pq$, qui génère un entier RSA, comme à sens unique. Elle permet d'établir le théorème suivant.

Théorème 5.1 *S'il existe une fonction à sens unique, alors, tout prédicat polynomial admet une preuve ZK algorithmique pour le problème NP associé.*

La preuve du théorème utilise en fait une fonction h définie sur le domaine $\{0,1\}^k \times \{B, R, V\}$ et qui a les propriétés suivantes :

i) Y est défini de manière unique à partir de $h(r, Y)$: si l'on préfère $h(r, Y) = h(r', Y')$ implique $Y = Y'$.

ii) Aucun test polynomial probabiliste ne distingue l'une de l'autre les trois distributions $\mathcal{D}_Y$, images de la fonction $r \rightarrow h(r, Y)$, où Y est B, R ou V.

Ainsi, pour tout algorithme polynomial probabiliste $\mathcal{A}$ dont la sortie est un unique bit, la quantité

$$\left|\Pr\left[\mathcal{A}(h(r, B) = 1 | y \stackrel{R}{\leftarrow} \{0,1\}^k\right] - \Pr\left[\mathcal{A}(h(r, R)) = 1 | x \stackrel{R}{\leftarrow} \{0,1\}^k\right]\right|$$

décroît plus vite que l'inverse de tout polynôme, et de même pour les autres quantités analogues. On démontre que l'existence d'une fonction à sens unique f implique l'existence d'une telle fonction h.

Preuve. (Indications sur la preuve du théorème 5.1.) Il suffit d'établir le résultat pour un problème NP-complet particulier par exemple le problème du coloriage d'un graphe à l'aide de trois couleurs, considéré dans la section 5.1.2. Tout revient donc à rendre mathématiquement correctes les preuves de correction et de zero-knowledge. Soit h une fonction définie sur le domaine $\{0,1\}^k \times \{B, R, V\}$, comme indiqué ci-dessus. On remplace la fonction d'engagement $e_u = H(\pi(c(u))||r_u)$ par $h(r_u, \pi(c(u)))$, où $r_u \stackrel{R}{\leftarrow} \{0,1\}^k$. Il est clair que l'on ne peut distinguer e_u de l'une ou l'autre des distributions $\mathcal{D}_Y$ considérées plus haut.

Correction. On ne donne pas en détail la preuve de correction. Sans être difficile ; l'argumentation est technique et une preuve analogue est proposée dans la section suivante. On part d'une machine $\tilde{P}$ qui interagit avec le vérifieur et conduit à l'acceptation avec probabilité non négligeable. Tout revient à choisir avec probabilité non négligeable, une exécution partielle de l'interaction qui mène à une situation où, après l'étape 1, la machine $\tilde{P}$ répond à toutes les questions du vérifieur. La propriété d'unicité *i)* de h, permet alors de définir, à partir des engagements $\{e_u | u \in \mathcal{V}\}$ de l'étape 1, un coloriage de G.

Simulation. La simulation se fonde, comme indiqué dans la section 5.1.2, sur l'anticipation de la question $\{u, v\}$ du vérifieur. On commence par observer qu'il est impossible de distinguer entre elles deux distributions qui sont des produits de $[E|$ distributions $\mathcal{D}_Y$. On introduit pour cela les distributions "hybrides" $\mathcal{D}_i$, où les i premières distributions proviennent du premier produit et les autres du second. Les hybrides extrêmes sont les deux distributions données et deux hybrides d'indices successifs ne diffèrent que sur une coordonnée : un test statistique ne les distingue donc qu'avec une probabilité $\varepsilon(k)$ négligeable. En utilisant l'inégalité triangulaire, on voir qu'un test polynomial probabiliste ne distingue les extrêmes qu'avec probabilité au plus $|E| \cdot \varepsilon(k)$, qui reste négligeable. Ceci montre déjà que l'étape 1 de la simulation est indistinguable de l'étape 1 de l'interaction "réelle" du prouveur avec $\tilde{V}$.

Tout revient alors à comprendre pourquoi la question du vérifieur consécutive aux données simulées, soit q_s, est égale à la question qui a été anticipée q_a, avec probabilité $\simeq \frac{1}{|E|}$. Soit $\mathcal{T}$ le test probabiliste $q_s \stackrel{?}{=} q_a$. Ce test ne peut distinguer la distribution simulée, créée par q_a et les engagements $\{e_u | u \in \mathcal{V}\}$, du produit de la distribution de q_a et de $|E|$ distributions du type $\mathcal{D}_Y$. Dans le second cas, q_s et q_a sont des variables indépendantes et donc le test est satisfait avec probabilité $\frac{1}{|E|}$. Dans la simulation, la probabilité diffère donc de $\frac{1}{|E|}$ par une quantité négligeable.

On arrive ainsi à l'étape 3 au bout de $|E|$ essais environ en moyenne, en ayant produit

1. à l'étape 1 une distribution indistinguable du produit de $|E|$ distributions du type $\mathcal{D}_Y$;
2. à l'étape 2, ce qui résulte du programme de $\tilde{V}$;
3. à l'étape 3, des aléas $\pi(c(u))$, $\pi(c(v))$, r_u, r_v, indépendants de la distribution à l'étape 1 et fournissant une réponse correcte pour l'étape 4.

On a ainsi simulé une répétition et on peut passer à la suivante. Les ε obtenus en passant d'une distribution à une autre ne font que s'ajouter et, comme le nombre de répétitions est polynomial, le total reste négligeable. □

5.3 Preuves ZK d'identité

On va maintenant présenter des applications du ZK suffisamment efficaces pour être applicables dans la pratique. On note en effet que le nombre de répétitions requis par la preuve zero-knowledge de la section précédente est gigantesque. Il n'est de plus pas facile de créer des instances garanties difficiles du problème de coloriage, même si ce dernier est NP-complet.

5.3.1 Racines carrées modulo un entier RSA

Soit n un entier RSA de k bits, $n = pq$, comme défini dans la section 1.4.2. Pour qui connaît la factorisation de n, l'extraction de racines carrées modulo n se ramène, à l'aide du théorème des restes chinois, au même problème modulo respectivement p et q. Or ce problème est facile : si p est congru à 3 modulo 4 et si x est un carré modulo n, alors $x^{\frac{p+1}{4}} \bmod n$ est une racine carrée de x. On a en effet

$$\left(x^{\frac{p+1}{4}}\right)^2 = x^{\frac{p-1}{2}} \cdot x \bmod n$$

et $x^{\frac{p-1}{2}}$, qui est le symbole de Legendre, vaut 1. Un algorithme polynomial probabiliste à peine plus compliqué, résout le problème pour n'importe quel p. On note que, puisqu'il y a deux racines carrées modulo p et q, le nombre de racines carrées modulo n est 4. Bien entendu, seule une proportion $1/4$ des entiers modulo n sont des carrés.

En sens inverse, il existe une réduction de la factorisation au problème des racines carrées : pour factoriser n, on choisit au hasard y et on passe la valeur $x = y^2 \bmod n$ à l'oracle qui calcule les racines carrées. Soit z la réponse de l'oracle. On a

$$(z - y)(z + y) = 0 \bmod n$$

Si z est différent de $\pm y$ modulo n, alors le pgcd de $z+y$ et de n ne vaut ni 1 ni n. L'entier n a ainsi été factorisé. Pour conclure, on note que le résultat du calcul de l'oracle ne dépend que de la donnée qui lui est fournie et pas de celle des quatre racines carrées qui ont pu la produire : la probabilité que z soit différent de $\pm y$ modulo n est donc exactement $1/2$.

Le problème de l'extraction de racines carrées est donc équivalent à la factorisation de n. On peut alors le considérer comme hors de portée pour un choix de paramètres convenable.

5.3.2 Le schéma d'identification de Fiat-Shamir

Le schéma de Fiat-Shamir permet d'assurer un service d'authenticité — basé sur la possession d'un secret — sans révéler la moindre information sur le secret. Le service est fondé sur l'utilisation d'un entier RSA n dont la factorisation est gardée secrète. Chaque utilisateur est muni d'une clé publique $\mathsf{pk} = x$ et d'une clé secrète $\mathsf{sk} = y$ qui est une racine carrée de x modulo n. Pour s'identifier afin d'accéder à une ressource, un utilisateur exécute une interaction dans laquelle il joue le rôle du prouveur. Comme dans la section 5.1.2, il s'agit de la répétition de quatre étapes successives d'interaction :

1. Le prouveur choisit aléatoirement un élément r de $\mathbb{Z}_n^\star$, calcule $e = r^2 \bmod n$ et transmet au vérifieur la valeur de e.
2. Le vérifieur choisit aléatoirement un bit b et le transmet au prouveur.
3. Le prouveur révèle au vérifieur $u = ry^b \bmod n$.
4. Le vérifieur teste l'égalité $u^2 = ex^b \bmod n$.

Les étapes ci-dessus 1 à 4 sont répétées un nombre suffisant t de fois. A l'issue de ces répétitions, le vérifieur accepte si tous les tests réalisés aux étapes 4 ont reçu une réponse positive ; dans le cas contraire, le vérifieur rejette la preuve. Les étapes 1 à 4 sont résumées dans la figure 5.2.

5.3.3 Preuve du schéma de Fiat-Shamir

Il est clair qu'un prouveur qui dispose d'une racine carrée de x et qui suit exactement le protocole sait répondre correctement aux questions du vérifieur. Reste donc à montrer les propriétés de correction et de zero-knowledge. On fait l'hypothèse que le nombre t de répétitions croît plus vite que toute fonction linéaire en $\log k$. On rappelle que k, le paramètre de sécurité, est le nombre de bits de n, soit $\log(n)$.

Correction. On examine d'abord la correction. Soit $\tilde{P}$ une machine qui interagit avec le vérifieur et conduit à l'acceptation avec probabilité non négligeable. L'exécution des interactions entre les deux machines dépend des aléas respectifs de $\tilde{P}$ et du vérifieur, qu'on

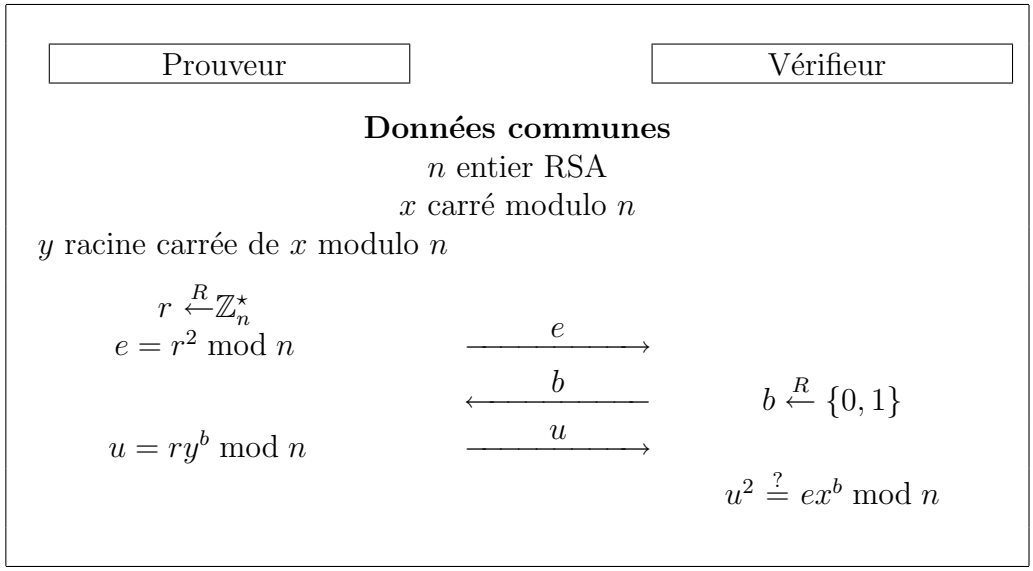


FIG. 5.2 – Preuve ZK de Fiat-Shamir

note ω et β. On note $T_i(\omega, \beta)$, $i \leq t$, la variable aléatoire qui vaut 1 si, après i répétitions des quatre étapes élémentaires représentées sur la figure 5.2, le vérifieur a toujours accepté. On note que T_i ne dépend que de segments initiaux des aléas ω et β, qu'on note respectivement ω_i et β_i. On pose

$$\varepsilon_i = \Pr\left[T_i = 1 \,|\, T_{i-1} = 1\right]$$

où T_0 vaut toujours 1 par convention.

Lemme 5.1 *Il existe un indice i tel que*

$$\varepsilon_i \geq \frac{3}{4}.$$

Preuve. En effet, la probabilité ε que le vérifieur accepte est le produit des ε_i. Si la conclusion du lemme n'est pas vraie, on a

$$\varepsilon = \prod_{i=1}^{t} \varepsilon_i < \left(\frac{3}{4}\right)^t$$

qui, puisque t croît plus vite que toute fonction linéaire en $\log k$, est une quantité négligeable, contrairement aux hypothèses. □

En prenant des aléas de type ω_{i+1} et β_i, on peut exécuter l'interaction de $\tilde{P}$ et du vérifieur jusqu'à la première étape de la $i+1$-ième répétition. On dit que le couple (ω_{i+1}, β_i) est *favorable* si $T(\omega_{i+1}, \beta_{i+1}) = 1$ pour tout prolongement β_{i+1} de β_i. Autrement dit, si le prouveur répond correctement aux deux questions que le vérifieur peut poser à l'étape 2.

Lemme 5.2 *Il existe un indice i tel que l'ensemble des couples favorables (ω_{i+1}, β_i) soit de probabilité non négligeable.*

Preuve. Soit (ω_{i+1}, β_i) un couple tel qu'on ait $T_i(\omega_{i+1}, \beta_i) = 1$. Si ce couple n'est pas favorable, la proportion des questions du vérifieur conduisant à l'acceptation est $\leq \frac{1}{2}$. La contribution des couples non favorables à la probabilité conditionnelle ε_i est donc $\leq \frac{1}{2}$. Comme cette probabilité est $\geq \frac{3}{4}$, la contribution des couples favorables est au moins égale

à $\frac{1}{4}$. On en déduit que la probabilité de l'ensemble des couples favorables ne peut être négligeable. On note de plus qu'en prenant i, ω_{i+1} et β_i au hasard, on a une probabilité non négligeable d'aboutir à un couple favorable. □

Soit finalement (ω_{i+1}, β_i) un couple favorable obtenu comme indiqué ci-dessus : soit e l'engagement transmis par $\tilde{P}$ à la première étape de la $i+1$-ième répétition. On note u_0 et u_1 les réponses fournies par $\tilde{P}$, lors de l'étape 3, aux questions respectives $b = 0$ et $b = 1$ du vérifieur. On a :

$$u_0^2 = e \bmod n$$

$$u_1^2 = ex \bmod n$$

Par suite :

$$(u_1 u_0^{-1} \bmod n)^2 = x \bmod n$$

On a bien ainsi extrait une racine carrée de x.

Simulation. La simulation se fonde, comme toujours, sur l'anticipation de la question b du vérifieur. Compte-tenu de cette anticipation b_a, le simulateur choisit au hasard u dans $\mathbb{Z}_n^\star$ et calcule l'engagement qu'il transmet à l'étape 1 par $e = u^2(x^{b_a})^{-1} \bmod n$. On fait ensuite "tourner" la machine $\tilde{V}$. Si la question qu'elle pose a été correctement anticipée, on passe aux étapes 3 et 4. Sinon, on réitère la simulation en réalisant une nouvelle anticipation. Le processus est répété autant de fois qu'il y a de répétitions, soit t. On va montrer que la simulation est parfaite et que le nombre moyen de répétitions du simulateur est $2t$.

On observe d'abord que, si x est un carré, les deux distributions de e définies par $e = u^2(x^b)^{-1} \bmod n$, pour $b = 0, 1$, sont précisément identiques et égales à la distribution uniforme sur les carrés de $\mathbb{Z}_n^\star$. Est également identique la distribution créé par $e = u^2(x^{b_a})^{-1} \bmod n$, lorsque u et b_a sont tirés au hasard. Les distributions produites à l'étape 1 lors de la simulation et lors d'une interaction "réelle" du prouveur avec $\tilde{V}$ sont donc les mêmes.

De ce qui précède, on déduit que la question b obtenue en faisant tourner le vérifieur est une variable aléatoire indépendante de b_a. On a donc $b = b_a$ avec probabilité $1/2$. On arrive ainsi à l'étape 3 au bout de 2 essais en moyenne en ayant produit

1. à l'étape 1 une distribution parfaitement identique à la distribution uniforme sur les carrés de $\mathbb{Z}_n^\star$;
2. à l'étape 2, ce qui résulte du programme de $\tilde{V}$;
3. à l'étape 3, un élément aléatoire de $\mathbb{Z}_n^\star$ fournissant une réponse correcte pour l'étape 4.

On a ainsi simulé une répétition et on peut passer à la suivante.

5.3.4 Le contexte cryptographique

Une des particularités de la cryptographie est la nécessité de mettre en place une infrastructure des clés. En d'autres termes, il s'agit de générer des instances génériquement difficiles des problèmes qui sous-tendent les schémas cryptographiques. C'est le rôle de l'algorithme de génération de clés. Formellement, on peut le voir comme un algorithme polynomial probabiliste prenant en entrée le paramètre de sécurité seul. Dans le cas du Fiat-Shamir par exemple, cet algorithme va produire un entier RSA $n = pq$ et, pour chaque utilisateur U, un couple $(\mathsf{pk}_U, \mathsf{sk}_U)$, la clé publique pk_U étant le carré de la clé secrète sk_U.

On peut choisir au hasard sk_U dans $\mathbb{Z}_n^\star$ et publier les pk_U correspondants, en authentifiant la publication par un mécanisme cryptographique, assurant la non-répudiation.

Une autre particularité de la cryptologie est la présence de l'adversaire. Il s'agit ici d'une MTPPI, qui prend en entrée la seule liste des clés publiques, et que l'on suppose capable

- d'exécuter des interactions avec l'un quelconque des utilisateurs, en jouant le rôle du vérifieur ;
- d'exécuter des interactions avec un vérifieur, en se faisant passer pour l'un quelconque des utilisateurs.

Il s'agit de démontrer qu'un tel adversaire n'a qu'une probabilité négligeable de parvenir à une acceptation dans une interaction du second type. Il est possible de démontrer que l'existence d'un adversaire qui est accepté avec une probabilité non négligeable implique celle d'une machine qui calcule des racines carrées (et donc factorise) avec probabilité non négligeable. Il est plus instructif de renoncer au point de vue asymptotique pour un résultat "exact" :

Théorème 5.2 *Soit $\mathcal{A}$ un adversaire probabiliste qui est accepté avec probabilité $\varepsilon > \frac{1}{2^t}$ dans un système de Fiat-Shamir à t répétitions basé sur un entier RSA de k bits, après K interactions avec des utilisateurs. Soit T le temps de calcul de $\mathcal{A}$ et soit τ le temps total des quatre étapes d'une interaction élémentaire. Alors, il existe une machine qui factorise les entiers RSA de k bits avec probabilité ε' et dont le temps de calcul moyen est T', avec*

$$T' = T + (2K+2)t\tau$$

$$\varepsilon' \geq 2\varepsilon(\varepsilon^{\frac{1}{t}} - \frac{1}{2})$$

Preuve. Pour établir le théorème on commence par supposer $K = 0$. La machine $\mathcal{A}$ joue alors le rôle de $\tilde{P}$ dans la preuve de correction. Il s'agit donc de rendre optimales les évaluations des probabilités dans la construction de l'extracteur, à laquelle on va se référer librement pour les notations. On pose $\varepsilon = (\frac{1}{2}+\delta)^t$. On en déduit que l'une des probabilités conditionnelles ε_i est $\geq (\frac{1}{2}+\delta)$. Soit α la proportion de couples favorables (ω_{i+1}, β_i) parmi les couples tels que $T_i(\omega_{i+1}, \beta_i) = 1$. On a $\varepsilon_i \leq \alpha + \frac{1}{2}(1-\alpha)$ et donc $\alpha \geq 2\delta$. Finalement, les aléas (ω, β), dont une restriction convenable est un couple favorable, sont en probabilité au moins $2\delta\varepsilon$. Une fois un tel aléa choisi, on extrait la racine carrée de pk_U en temps $2t\tau$.

Pour passer au cas général, on se donne un carré x de $\mathbb{Z}_n^\star$ dont on cherche à extraire la racine. On génère des clés publiques pour les utilisateurs de la forme $\mathsf{pk}_U = xv^2 \bmod n$ avec $v \overset{R}{\leftarrow} \mathbb{Z}_n^\star$. On fait ensuite tourner l'adversaire $\mathcal{A}$. La difficulté est qu'on ne connaît pas les clés secrètes des utilisateurs et qu'on ne peut donc interagir avec $\mathcal{A}$ lorsque ce dernier joue le rôle du vérifieur. On est toutefois précisément dans le contexte de la simulation ZK et donc, on procède à cette simulation, ce qui nécessite en moyenne l'exécution de $2Kt$ répétitions des quatre étapes de base. On se trouve ainsi ramené au cas précédent et on extrait ainsi une racine de pk_U et donc de x. □

5.3.5 Le Fiat-Shamir à plusieurs secrets

Dans le schéma de Fiat-Shamir, l'ensemble des communications représente environ $2tk$ bits, où t est le nombre de répétitions et k le nombre de bits de l'entier RSA n. On recommande aujourd'hui $k \geq 1024$, ce qui, si l'on veut borner la probabilité de succès d'un faussaire à 2^{-20}, amène à un total de 5 kilo-octets. Pour diminuer la bande passante, Fiat et Shamir ont proposé un schéma à ℓ secrets y_i, chacun d'eux étant la racine carrée d'une

quantité publique. La figure 5.3 représente, sur le modèle habituel, les quatre étapes d'un Fiat-Shamir à plusieurs secrets. Ces étapes sont répétées t fois.

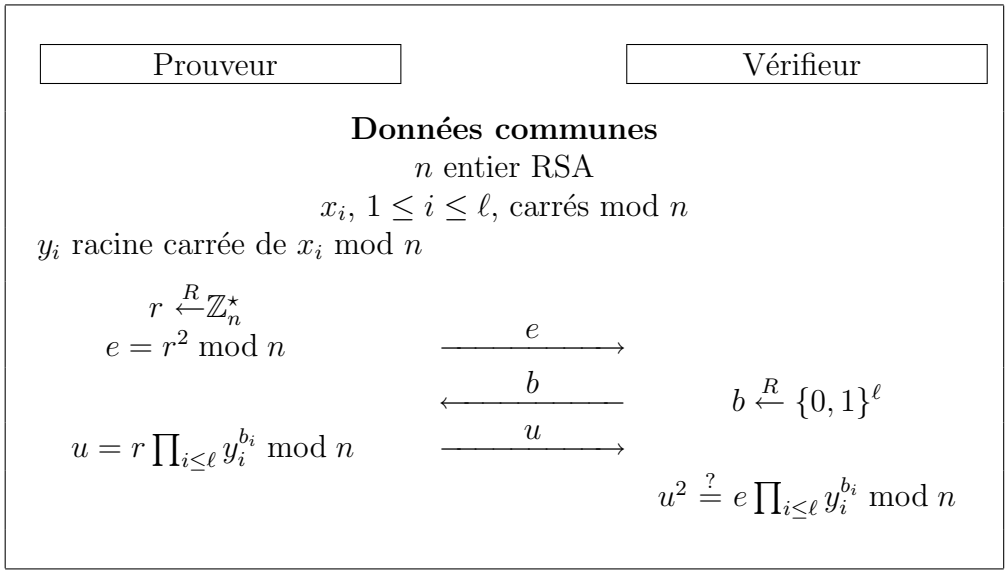


FIG. 5.3 – Fiat-Shamir à plusieurs secrets

Le schéma de Fiat-Shamir à plusieurs secrets constitue une preuve zero-knowledge de connaissance de racines carrées des ℓ éléments $x_1, \cdots, x_\ell$. La preuve de simulation nécessite l'hypothèse $\ell = \mathcal{O}(\log k)$, car on doit anticiper une valeur dans $\{0,1\}^\ell$. La preuve de correction recherche, pour chaque indice i, une situation où le prouveur sait répondre à deux questions égales sur tous les indices sauf i. On note que la version "exacte" de l'extracteur ne produit l'ensemble des racines carrée des x_i que si l'on part d'une machine $\tilde{P}$ qui est acceptée avec probabilité $\varepsilon > \frac{1}{2^t}$. Il semble donc qu'on n'ait rien gagné sur le Fiat-Shamir ordinaire. Toutefois, si l'on cherche seulement à obtenir une situation où le prouveur sait répondre à deux questions *distinctes*, alors on extrait une racine d'un produit non trivial des x_i, dès que $\varepsilon > \frac{1}{2^{t\ell}}$. On laisse au lecteur le soin d'énoncer et de prouver l'analogue du théorème 5.2, en utilisant cette propriété. En pratique, à sécurité "constante", le Fiat-Shamir à plusieurs secrets divise le nombre de répétitions par ℓ.

5.3.6 Le schéma de Guillou-Quisquater

Le schéma de Guillou-Quisquater (voir [5]) est une preuve de connaissance d'une racine e-ième d'un élément de $\mathbb{Z}_n^\star$, où e un nombre premier. La figure 5.4 représente, sur le modèle habituel, les quatre étapes d'un Guillou-Quisquater. Ces étapes sont répétées t fois.

On laisse le détail des preuves au lecteur. La preuve de simulation nécessite l'hypothèse $\ell = \mathcal{O}(\log k)$, car on doit anticiper une valeur dans $\{0,1\}^\ell$. La preuve de correction recherche une situation où le prouveur sait répondre à deux questions distinctes b et b', $b < b'$. On a alors, en notant u et u' les réponses respectives à ces questions

$$u^e = cx^b \bmod n$$

$$(u')^e = cx^{b'} \bmod n$$

ce qui donne

$$(u'u^{-1} \bmod n)^e = x^{b'-b} \bmod n$$

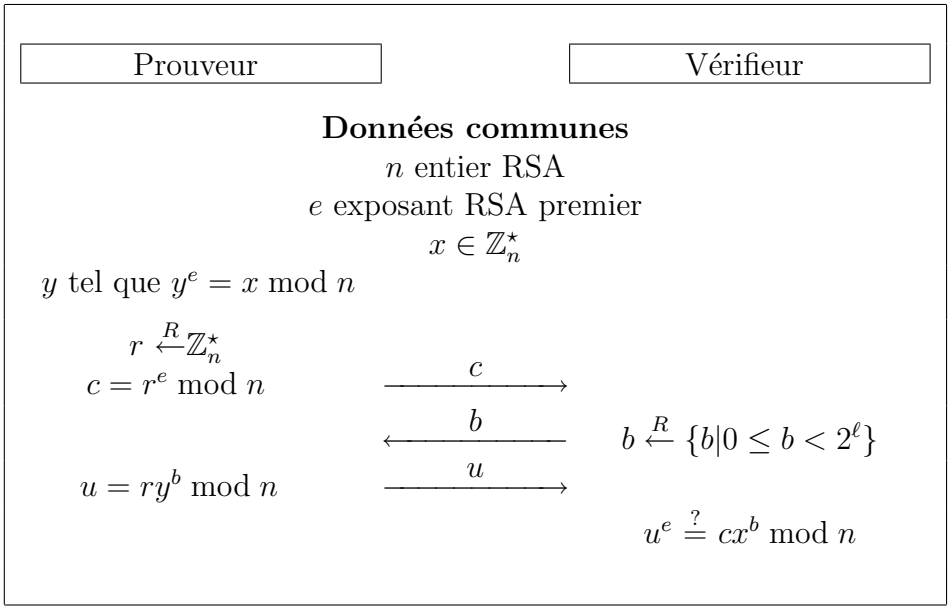


FIG. 5.4 – Schéma de Guillou-Quisquater

On fait l'hypothèse que $e > 2^\ell$. Les entiers e et $b' - b$ sont alors premiers entre eux et l'algorithme d'Euclide étendu permet de calculer les coefficients de Bézout a et m tels que $ae + m(b' - b) = 1$. Il vient

$$x = x^{ae+m(b'-b)} = (x^a(u'u^{-1})^m \bmod n)^e \bmod n$$

On a ainsi extrait une racine e-ième de x.

Le schéma de Guillou-Quisquater nécessite une bande passante analogue à celle de Fiat-Shamir. Il est un peu plus gourmand en capacité de calcul à cause des exponentielles modulaires. En revanche, il n'a besoin que d'un seul secret. Il est tentant de ne faire qu'une répétition et d'augmenter ℓ à la place. Cependant, on ne peut plus obtenir ainsi un schéma zero-knowledge.

Bibliographie

[1] U. Feige and A. Fiat and A. Shamir. Zero-Knowledge Proofs of Identity, *J. Cryptology*, 1, 1988, 77–94.

[2] A. Fiat and A. Shamir. How to prove yourself : Practical solutions to identification and signature problems, *Proceedings of Crypto 86*, Lecture Notes in Computer Science 263, 181–187.

[3] M. Garey & D. S. Johnson. *Computers and Intractability : A Guide to the Theory of NP-Completeness*, Freeman, 1979.

[4] S. Goldwasser, S. Micali and C. Rackoff. The knowledge complexity of interactive proof systems, *Proc. 17th ACM Symp. Theory of Computing*, (1985), 291–304.

[5] L.S. Guillou and J.-J. Quisquater. A practical zero-knowledge protocol fitted to security microprocessors minimizing both transmission and memory, *Proceedings of Eurocrypt 88*, Lecture Notes in Computer Science 339, 123–128.

[6] J. Stern A new paradigm for public key identification, *IEEE Trans. Inform. Theory*, 42 (6), 1996, 1757–1768.