École normale supérieure
LIENS – École Doctorale de Sciences Mathématiques de Paris Centre

Sur Quelques Méthodes Algébriques et Statistiques en Cryptanalyse

THÈSE D'HABILITATION

présentée pour l'obtention du

Diplôme d'Habilitation à Diriger des Recherches de l'École normale supérieure

(Spécialité Informatique)

par

Pierre-Alain Fouque

Soutenue publiquement le 10 décembre 2010 devant le jury composé de

Henri Gilbert ..... *Rapporteur*
Louis Goubin ..... *Rapporteur*
Bart Preneel ..... *Rapporteur*
Jean-Charles Faugère ..... *Examinateur*
Antoine Joux ..... *Examinateur*
David Pointcheval ..... *Examinateur*
Adi Shamir ..... *Examinateur*
Jacques Stern ..... *Directeur*

Travaux effectués au sein de l'Équipe de Cryptographie
du Laboratoire d'Informatique de l'École normale supérieure

Remerciements

Je remercie Jacques pour avoir encadré mes recherches depuis ma thèse et pour m'avoir fait découvrir le domaine passionnant de la cryptographie multivariée et ses nombreux schémas avec lesquels nous avons beaucoup joués.

Je remercie également Henri, Louis, Bart, Jean-Charles, Antoine, David et Adi pour l'honneur qu'ils me font de faire parti de ce jury. Merci à Henri, Louis et Bart d'assurer la lourde tâche de rapporter ce mémoire. Je remercie Adi d'avoir accepté d'assister à ma soutenance dès son retour d'Asiacrypt et pour nos riches échanges lors de nos travaux communs. Son enthousiasme sans relâche et sa mémoire sans faille de tous ses travaux continuent de m'impressionner. Je remercie David qui gère l'Équipe de Cryptographie en nous faisant une grande confiance sur nos sujets de recherche et pour nos nombreuses collaborations dans le domaine des preuves de sécurité. Ces travaux n'apparaissent pas dans ce document, mais ils représentent une partie importante de mes recherches. Je remercie Antoine, pour m'avoir fait découvrir de nouveaux domaines de recherche et pour le partage de ses nombreuses idées. Enfin, je remercie Jean-Charles qui me fait le plaisir de participer à ce jury.

Je remercie aussi Guillaume Poupard et Frédéric Valette pour l'ambiance inoubliable qu'ils ont créée au Labo Crypto de la DCSSI pendant les deux ans que j'y ai passé après ma thèse, et pour nos collaborations fructueuses et toujours joyeuses, en particulier dans le domaine des attaques par canaux auxiliaires.

Je remercie toutes les personnes de l'Équipe de cryptographie de l'École normale supérieure. Merci Michel, pour nos discussions et l'organisation de PKC qui nous a bien occupée cette année, et ceux qui ont relu ce manuscrit, Phong, avec qui je partage le cours de Cryptanalyse au MPRI, et Damien, pour nos échanges au sujet de différents exercices d'Algorithmique et de Cryptographie. Ensuite, je remercie tous mes coauteurs et plus particulièrement mes étudiants en thèse, passés et actuels, Sébastien, Gilles, Gaëtan, Charles, Delphine, Jérémy et Patrick avec lesquels je prends plaisir à travailler, et ceux avec qui j'ai eu le plaisir de collaborer, Frédéric Muller, Sébastien Kunz-Jacques, Vivien, Éric, Pierrick, Reynald, Denis, Ludovic, Céline, Orr et Mehdi. Enfin, je remercie Joëlle, Valérie, Isabelle et Michèle et Lise-Marie, qui nous aident quotidiennement dans l'organisation de la Recherche et de l'Enseignement ainsi que le Service de Prestation Informatique.

Enfin, merci à ma famille pour tout ce que je lui dois, et à Gwenaëlle, Thibault et Cléane pour leur amour, leurs encouragements et la compréhension dont ils ont fait preuve pendant la rédaction de ce mémoire.

Table des matières

Première partie

Sur Quelques Méthodes Algébriques et Statistiques en Cryptanalyse

Introduction

La cryptanalyse est l'*art de déchiffrer les messages secrets*. La cryptanalyse a eu un apport essentiel au cours des événements historiques : elle est au cœur de moments décisifs comme l'entrée en guerre des États-Unis lors de la Première Guerre mondiale suite au décryptement [1] du télégraphe Zimmermann [173, 274], ou lors des batailles d'Angleterre ou de l'Atlantique durant la Seconde Guerre mondiale suite au décryptement par le service du Chiffre britannique de plusieurs messages allemands, ou encore lors de la mort de Marie Stuart [173, 274].

Au cours de la Seconde Guerre mondiale, la cryptanalyse s'est considérablement développée sous l'influence de nombreux mathématiciens rassemblés à Bletchley Park, dont le plus célèbre est sans doute Alan Turing. Ces mathématiciens et cryptographes ont découvert des failles dans l'utilisation de certaines machines de chiffrement, comme la désormais célèbre ENIGMA. Ces failles leur ont permis de réduire le nombre de clés possibles : ils ont tout d'abord réalisé des machines, appelées « bombes », qui simulaient ENIGMA et permettaient de tester exhaustivement les clés restantes. La cryptanalyse du code de Lorenz, utilisé pour chiffrer les communications militaires du Haut-Commandement allemand, a nécessité l'invention d'une machine plus puissante et généraliste, capable d'exécuter des séquences d'instructions prédéfinies. Ce premier ordinateur a été conçu suite aux travaux théoriques d'Alan Turing. Ce dernier a défini en 1936 les machines de Turing comme modèle de calcul et qui permet de modéliser les algorithmes. Il a aussi défini les machines universelles, capables de simuler toutes les machines de Turing et donc tous les algorithmes, modélisant ainsi les ordinateurs. La machine Colossus a été construite en 1943 par Tommy Flowers à Bletchley Park pour décrypter le code Lorenz. Elle était capable d'effectuer 5000 opérations par seconde et décrypter un texte de 3000 caractères en 13 minutes [64]. L'informatique a donc née de la cryptanalyse pour laquelle elle reste un outil indispensable. Toutefois, la cryptanalyse utilise également des outils mathématiques extrêmement variés et de plus en plus complexes. Elle se positionne donc à la frontière de ces deux disciplines.

Les premières techniques mathématiques utilisées en cryptanalyse ont été les statistiques. Elles ont permis d'attaquer les systèmes de chiffrement anciens qui transformaient le texte clair au moyen d'une simple substitution des lettres. Comme les fréquences d'apparition des lettres de la langue d'origine sont invariantes par cette transformation, les mêmes biais statistiques apparaissent dans le texte chiffré et dans le texte clair. L'analyse de fréquence, développée par Al-Kindi au 9ième siècle, a été utilisée pour attaquer les systèmes construits à l'aide de substitutions. Cette analyse repose sur la distribution d'apparition des lettres, non-uniforme mais connue dans chaque langue. Par exemple en français, la lettre `e` a la plus grande fréquence d'apparition, la lettre `z` la plus faible. Considérons une substitution très simple, appelée chiffrement par décalage, qui consiste

1. Le décryptement est le résultat d'une cryptanalyse : retrouver le message clair sans utiliser la clé de déchiffrement.

à remplacer une lettre par une lettre située d lettres plus loin dans l'alphabet, de façon cyclique. Pour retrouver le décalage d, il suffit de repérer quelle lettre apparaît le plus fréquemment dans le texte chiffré. Si le texte est suffisamment long, on établit ainsi la correspondance entre cette lettre chiffrée et la lettre initiale. Par exemple, pour un texte clair en français, si `k` est la lettre la plus fréquente du texte chiffré, elle correspond à la lettre `e` en clair et le décalage d est le nombre de lettres entre `e` et `k`. Tout le texte peut alors être décrypté entièrement.

Puis, des techniques plus avancées, comme l'indice de coïncidence ont été développées pour attaquer des systèmes plus compliqués, comme le système de chiffrement de Vigenère. Ce système est dit polyalphabétique, car une lettre peut être chiffrée par plusieurs autres. Il a été proposé par Blaise de Vigenère au 16ième siècle et est resté inattaqué jusqu'au 19ième siècle [173, 274]. Ce système permet, à partir d'un mot de passe, d'utiliser pour chacune de ses lettres un chiffrement par décalage. L'indice de coïncidence fournit un test statistique sur le message chiffré qui permet de déduire la longueur de la clé. Une fois que la longueur de la clé est connue, il est facile de retrouver le mot de passe car il s'agit de plusieurs chiffrements par décalage en parallèle.

Au début de la Seconde Guerre mondiale, les plans de la machine Enigma utilisée par les armées allemandes n'étaient pas complètement connus des Alliés. Le fonctionnement de la version commerciale était connu, mais les rotors entre les versions étaient différents. Ces rotors étaient composés d'une partie fixe, sur laquelle était inscrit un alphabet, et d'une partie en mouvement, sur laquelle était inscrite une substitution fixe. À chaque chiffrement d'une lettre, la partie en mouvement augmentait d'une position, ce qui modifiait la substitution. Il y avait trois ou cinq rotors suivant les versions et une fois que le premier rotor avait effectué une rotation complète, il entraînait le second d'une position et ainsi de suite. Ainsi, au bout de $(26)^3$ ou $(26)^5$ lettres, la machine revenait dans sa configuration initiale. La clé de la machine Enigma était composée de la position initiale des rotors mais pas de la valeur de la substitution fixe, et d'une permutation initiale et finale, composée de sept transpositions. Pour retrouver la valeur des rotors, les services du Chiffre polonais ont fait appel à l'algèbre en étudiant la longueur des cycles des permutations [274]. Le recours à des méthodes algébriques pour étudier la sécurité des systèmes cryptographiques va alors s'accroître car les systèmes vont utiliser de plus en plus d'algèbre principalement avec l'arrivée de la cryptographie à clé publique.

La cryptanalyse moderne, depuis l'invention de la cryptographie à clé publique et du système de chiffrement par bloc Data Encryption Standard (DES) dans les années 1970, utilise l'algèbre, les statistiques et l'algorithmique. Pour analyser la sécurité d'un schéma, il est cependant nécessaire de définir les buts de l'adversaire et les moyens qu'il peut mettre en œuvre.

Pour un système de chiffrement, les buts de l'adversaire peuvent être : retrouver la clé secrète ou privée, retrouver une clé équivalente à cette clé, déchiffrer un message donné, distinguer le chiffrement d'un message d'une suite aléatoire. Pour un schéma d'authentification ou de signature, l'adversaire peut vouloir retrouver la clé secrète, une clé équivalente, ou contrefaire une signature. Enfin, pour les fonctions de hachage, les buts de l'adversaire sont de trouver efficacement des collisions, des préimages ou des secondes préimages. Afin de résoudre ces buts, l'adversaire a plusieurs moyens. Pour attaquer un système de chiffrement, l'adversaire peut avoir uniquement un message chiffré, connaître des couples (message clair ; message chiffré), avoir accès temporairement à une machine de chiffrement ou de déchiffrement. Pour une fonction d'authentification ou de signature,

l'adversaire peut connaître certains couples message-signature pour des messages connus ou choisis. L'association d'un but et d'un moyen définit un problème cryptographique que le cryptanalyste va devoir résoudre.

Ce mémoire présente quelques méthodes algébriques et statistiques pour résoudre les problèmes cryptographiques.

La cryptanalyse des systèmes à clé publique emploie essentiellement des méthodes algébriques car il existe naturellement des relations algébriques entre la clé publique et la clé privée. De même, le message chiffré s'écrit sous la forme d'une équation mêlant le message, vu comme un entier, et la clé publique. Par exemple, dans le système RSA [257] inventé par Rivest, Shamir, et Adleman en 1977, un entier produit de deux nombres premiers de grandes tailles, $N = p \cdot q$, définit la structure mathématique d'anneau dans laquelle vont avoir lieu les opérations. Les clés publique e et privée d sont deux entiers vérifiant l'équation

$$e \times d = 1 \bmod (p-1)(q-1),$$

et le chiffrement d'un message, le transforme en un entier m et calcule c tel que,

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

L'entier c représente le message chiffré. Le problème RSA consiste à retrouver l'entier m correspondant à un chiffré c choisis aléatoirement, connaissant les entiers N et e. Ce problème est toujours conjecturé comme étant difficile puisque personne n'a proposé d'algorithme efficace pour le résoudre. Cependant, même si certaines instances de ce problème apparaissent difficiles, toutes les instances ne le sont pas forcément. Coppersmith a proposé en 1996 un algorithme s'exécutant en temps polynomial quand les instances cachent des messages où la taille de la partie inconnue du message est petite. Les motivations pour étudier ces instances particulières viennent des schémas proposés en pratique car la cryptographie à clé publique est employée pour chiffrer des clés de session de petite taille, typiquement 128 bits. Comme l'exposant publique e est souvent choisi égal à 3 pour des raisons de performances, le message recherché est bien inférieur à $N^{1/e}$, pour des entiers N de 1024 bits par exemple. Il est donc possible de retrouver ces clés de session pour des instances de cette forme.

Les attaques actuelles concernent des schémas dont la sécurité repose sur des instances particulières d'un problème difficile. C'est le cas par exemple des systèmes de chiffrement dont la clé publique est formée d'un système de polynômes en plusieurs variables. Le problème consistant à déchiffrer ces systèmes se ramènent au problème général de résoudre de tels systèmes. Les schémas de chiffrement sont construits en masquant des instances *faciles* à résoudre au moyen d'une clé secrète. De cette façon, le détenteur de la clé privée peut aisément inverser le système. Les concepteurs supposent que ces instances faciles, cachées au moyen de la clé secrète, sont aussi difficiles que les instances aléatoires ou les plus difficiles de ce problème. La tâche du cryptanalyste est alors de concevoir des algorithmes pour résoudre ces instances très particulières d'un problème difficile.

La cryptanalyse des systèmes symétriques repose principalement sur des techniques statistiques ou algébriques. L'adversaire recherche des éléments caractéristiques du schéma qui le distingue d'un chiffrement parfait, c'est-à-dire dans lequel le chiffré serait indistinguable d'une chaîne binaire aléatoire de même taille. Par exemple considérons un système de chiffrement par substitution et définissons l'estimateur connu sous le nom de l'*indice*

de coïncidence,

$$\sum_{a \in A} \frac{f_a(f_a - 1)}{n(n-1)}$$

où a représente une lettre de l'alphabet A, f_a la fréquence d'apparition de la lettre a dans le texte et n la longueur du texte. Comme les fréquences d'apparition des lettres dans les mots d'une langue sont différentes de lettres choisies uniformément, cet estimateur est différent pour un texte écrit dans une langue et pour un texte où les mots seraient des lettres choisies aléatoirement. Par exemple, pour la langue française, il est égal à $0,0778$, $0,0667$ pour l'anglais, et $0,0529$ pour le russe et $0,038$ pour un texte aléatoire. De plus, comme le chiffrement par substitution change chaque lettre dans l'alphabet clair par une autre lettre dans l'alphabet chiffré, il ne modifie pas globalement les fréquences d'apparition. Cet estimateur est donc *invariant* par ce système de chiffrement. Par conséquent, calculé sur le texte chiffré, il permet de définir un *distingueur* du système de chiffrement, c'est-à-dire un algorithme qui distingue le chiffrement d'un message, d'une suite aléatoire. Nous pouvons également remarquer qu'il peut aussi servir à connaître la langue dans laquelle est écrite le texte clair si le système de chiffrement est une substitution.

Les attaques statistiques reposent *in fine* sur la recherche exhaustive de parties de la clé en transformant ces distingueurs en algorithme retrouvant la clé secrète : le distingueur est un test d'arrêt repérant la bonne clé. C'est le cas des attaques par corrélation, développées par Siegenthaler en 1985, détecte les corrélations entre la suite chiffrante produite par un système de chiffrement par flot utilisant plusieurs registres et la suite produite par un sous-ensemble des registres [273]. Les attaques linéaires, développées par Matsui sur le DES en 1993 [211] à partir de travaux de Gilbert de ses coauteurs [276], utilisent le biais d'un bit calculé par une combinaison linéaire entre les bits du message, ceux du chiffré et ceux de la clé d'un système de chiffrement par bloc. Les attaques différentielles, développées par Biham et Shamir en 1990, exploitent la propagation des différences les plus probables quand deux messages ayant une différence fixée sont chiffrés [35]. Ces corrélations, ces biais ou ces différences les plus probables sont exploités comme test d'arrêt pendant l'attaque pour retrouver efficacement plusieurs bits de clés.

Afin de résister aux attaques linéaires ou différentielles, les systèmes de chiffrement ont incorporé des outils algébriques. C'est le cas du schéma de chiffrement AES [84], conçu par Daemen et Rijmen, qui a une description très algébrique grâce à laquelle il est possible de prouver certaines bornes inférieures sur les probabilités des attaques statistiques. La contrepartie est qu'il existe un distingueur, algorithme permettant de distinguer la fonction de chiffrement AES d'une permutation aléatoire, sur quelques tours qui repose sur des propriétés algébriques des éléments constituants ce chiffrement. De même, plusieurs attaques sur les systèmes de chiffrement par flot sont de nature algébrique. En effet, pour éviter les attaques par corrélation, le critère d'immunité contre ces attaques impose que le degré algébrique des fonctions booléennes soit petit. Les attaques algébriques sur les chiffrement par flot exploitent cette propriété.

Les systèmes cryptographiques sont aujourd'hui proposés avec des preuves de sécurité. Il est donc très difficile de les attaquer, mais les preuves de sécurité se font toujours dans un modèle d'attaque qui limite ce que peut faire l'adversaire. En cryptanalyse classique, les fonctions cryptographiques sont étudiées d'un point de vue mathématique comme une boîte noire, c'est-à-dire en donnant à l'attaquant des couples entrées/sorties de la fonction. En pratique, ces fonctions sont calculées dans les processeurs avec des algorithmes et l'étude des implémentations peut révéler des failles de sécurité. En 1996, Kocher a proposé

des attaques exploitant les propriétés *matérielles et logicielles* des systèmes de chiffrement comme la mesure du temps de calcul ou de la consommation de puissance du processeur. Ceci permet d'obtenir des informations supplémentaires sur la fonction calculée comme de savoir quelles instructions sont exécutées ou d'apprendre le contenu de variables internes au calcul. En utilisant des méthodes algébriques ou statistiques et ces informations supplémentaires, il est bien souvent possible de concevoir des attaques très efficaces retrouvant les éléments secrets. Ces attaques sont appelées *attaques par canaux auxiliaires.*

Dans ce mémoire, nous allons décrire quelques méthodes de cryptanalyse. La première partie présente des attaques algébriques sur les systèmes à clé publique où l'adversaire doit résoudre des systèmes linéaires avec solutions contraintes, des attaques sur des systèmes cryptographiques reposant sur la difficulté à résoudre des systèmes polynomiaux en plusieurs variables, et des distingueurs sur les systèmes cryptographiques à clé secrète. La seconde partie est consacrée aux attaques statistiques sur les systèmes symétriques. Tout d'abord, elle décrit des attaques génériques sur les systèmes de chiffrement, sur les fonctions de hachage et sur les codes d'authentification de message en utilisant des collisions internes dans les mécanismes. Puis, le problème LPN et différents algorithmes pour le résoudre sont présentés. Ces algorithmes correspondent à différentes attaques dans les systèmes de chiffrement par flot. Enfin, nous présentons une application des attaques différentielles en collisions sur les fonctions de hachage au schémas d'authentification. La dernière partie présente quelques attaques par canaux auxiliaires par mesure du temps, de la consommation de courant ou par fautes.

Chapitre 1

Méthodes algébriques

Dans ce chapitre, nous allons nous intéresser aux attaques cryptographiques exploitant les relations algébriques vérifiées par le mécanisme attaqué. Ces relations forment un système algébrique que doit résoudre le cryptanalyste.

En cryptographie asymétrique, il existe naturellement des relations algébriques entre la clé publique et la clé privée. De même, il existe des équations reliant le chiffré au message clair ou la signature au message à signer. Selon les schémas, les systèmes obtenus sont intrinsèquement différents : certains problèmes se ramènent à résoudre des systèmes d'équations linéaires pour lesquels les solutions sont contraintes, d'autres demandent de résoudre des systèmes polynomiaux contraints, en une ou plusieurs variables, et de degrés divers.

En cryptographie symétrique, il existe également des attaques algébriques contre les primitives. Nous verrons que la recherche de propriétés algébriques, de polynômes de petit degré, d'équivalents algébriques, peut permettre de distinguer des fonctions de chiffrement de fonctions idéales.

1.1 Système linéaire avec petites solutions

Dans cette section, nous allons voir que certains problèmes cryptographiques se ramènent à résoudre des systèmes linéaires pour lesquels les solutions sont petites. Dans certains cas, le système sera composé d'une *seule* équation.

Algorithme LLL. L'outil pour résoudre ce type de problème est l'algorithme LLL [8] qui permet de trouver un vecteur court dans un réseau. Un réseau est défini comme l'ensemble des points résultants des combinaisons linéaires à coefficients entiers d'un ensemble de vecteurs libres. Dans un réseau L de dimension n, c'est-à-dire possédant une base de n vecteurs, le plus court vecteur est de norme inférieure à $\sqrt{n}\det(L)^{\frac{1}{n}}$ d'après le théorème de Minkowski [62]. Une heuristique habituelle est de considérer que le vecteur le plus court d'un réseau typique est en fait de cette taille à une petite constante près.

L'algorithme LLL s'exécute en temps polynomial en la taille de la base d'entrée. Il retourne un vecteur v approchant le plus court vecteur à un facteur multiplicatif $2^{\frac{n-1}{2}}$ près. Différents algorithmes ont été proposés comme par exemple [265] par Schnorr et Euchner. La thèse d'habilitation de Phong Nguyen présente de façon très complète l'algorithmique des réseaux [224]. En pratique, l'algorithme LLL se comporte beaucoup mieux que la théorie ne le prédit. Après de nombreuses expérimentations, Nicolas Gama et Phong Nguyen ont conjecturé dans [132] que le facteur approximatif est de l'ordre de 1.0219^n en grande dimension, $n \geq 100$.

Le calcul du déterminant peut être complexe et donc l'estimation de la norme du vecteur le plus court peut s'avérer problématique. Dans ce cas, il est possible d'utiliser l'inégalité de Hadamard qui pose que $\det(L) \leq \prod_{i=1}^{n} \|b_i\|$, où les b_i sont les vecteurs de la base initiale.

En cryptanalyse, le vecteur-cible, lié algébriquement à la clé secrète, est de norme anormalement petite comparée à celle des vecteurs de base. L'algorithme LLL peut alors permettre de le retrouver. Les articles [170, 226] détaillent de nombreuses applications de la réduction de réseau à la cryptographie.

Application à la factorisation d'un module RSA connaissant certains bits d'un des facteurs. Soit $N = pq$ un module RSA de taille n. Dans [256], Rivest et Shamir supposent que les $n/3$ bits de poids forts de p sont connus et tentent de factoriser N en retrouvant les $n/6$ bits restants de p. Posons

$$p = p_1 2^{n/6} + p_0,$$

où p_0 représente les $n/6$ bits inconnus de p et p_1 les $n/3$ bits connus. Les $n/3$ bits de poids forts de q, représentés par q_1, peuvent alors être calculés de la façon suivante :

$$\left\lfloor \frac{N}{p_1 2^{n/6}} \right\rfloor = 2^{n/6} q_1 + \left(q_0 + p_0 \frac{q_1}{p_1} \right) \approx 2^{n/6} q_1.$$

En posant $X = N - p_1 q_1 2^{n/3}$, nous avons l'équation linéaire suivante, fonction des inconnues p_0, q_0 et $p_0 q_0$:

$$p_1 2^{n/6} \cdot q_0 + q_1 2^{n/6} \cdot p_0 + p_0 q_0 = X. \tag{1.1}$$

L'algorithme LLL permet de trouver les solutions à une telle équation. En effet, considérons le réseau L engendré par les lignes de la matrice suivante, où K est une constante.

$$\begin{pmatrix} p_1 2^{n/6} K & 1 & 0 & 0 \\ q_1 2^{n/6} K & 0 & 1 & 0 \\ -XK & 0 & 0 & 1 \\ K & 0 & 0 & 0 \end{pmatrix}.$$

En multipliant cette matrice à gauche par le vecteur $(q_0, p_0, 1, p_0 q_0)$, nous obtenons le vecteur du réseau

$$(K(p_1 2^{n/6} \cdot q_0 + q_1 2^{n/6} \cdot p_0 - X + p_0 q_0), q_0, p_0, 1) = (0, q_0, p_0, 1) \tag{1.2}$$

dont la norme est de l'ordre de $2^{n/6}$.

Ce vecteur-cible est particulièrement court par rapport aux autres vecteurs du réseau. Les vecteurs lignes du réseau sont de taille respective $K\sqrt{N}, K\sqrt{N}, K2^{2n/3}, K$. D'après le théorème de Minkowski, le vecteur le plus court du réseau est donc de norme de l'ordre de $\sqrt{4}\det(L)^{\frac{1}{4}} = 2|K|^{1/4}$. Pour que le vecteur (1.2) soit le plus court, il faut donc que $K^{1/4} \gg 2^{n/6}$. Il suffit donc de choisir $K \gg 2^{2n/3}$ pour retrouver les facteurs p et q avec le vecteur (1.2).

Application au Problème du sac-à-dos. L'algorithme LLL a été initialement utilisé en cryptanalyse pour attaquer les schémas construits à partir d'une instance du problème du sac-à-dos. Pour inverser ces systèmes de chiffrement et retrouver le message, le problème consiste à trouver une solution en $x = (x_1, \ldots, x_n) \in \{0,1\}^n$ de l'équation

$$\sum_{i=1}^{n} a_i x_i = b,$$

où b et les a_i sont des entiers. Un paramètre important des instances du problème du sac-à-dos est la densité d qui s'exprime par la quantité [158]

$$d = \frac{n}{\max_{1 \leq i \leq n} \log_2(a_i)}.$$

Pour $d < 1$, le problème possède heuristiquement au plus une solution. Ces instances sont alors utiles pour construire des schémas de chiffrement. Pour $d > 1$, plusieurs solutions sont possibles et ces instances permettent de construire des fonctions de hachage qui n'exigent pas d'être inversibles.

Considérons le réseau L de dimension $(n+1)$ engendré par les lignes de la matrice suivante où $K = \lceil \frac{1}{2}\sqrt{n} \rceil$:

$$\begin{pmatrix} 1 & 0 & 0 & \dots & 0 & Ka_1 \\ 0 & 1 & 1 & \dots & 0 & Ka_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 & Ka_n \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \dots & \frac{1}{2} & -Kb \end{pmatrix}.$$

En multipliant cette matrice par le vecteur $(x_1, \dots, x_n, 1)$, nous obtenons le vecteur-cible de L de coordonnées $(x_1 + \frac{1}{2}, \dots, x_n + \frac{1}{2}, 0)$ et de norme inférieure à $\sqrt{3n/4}$. La norme des vecteurs initiaux du réseau s'exprime en fonction de d comme $|K|2^{n/d}$. Il a été prouvé que le vecteur cible est, avec très forte probabilité, le vecteur le plus court quand $d < 0.94$ [79, 170]. Dans ce cas, l'algorithme LLL permet de résoudre ces instances en petite dimension pour $n \leq 110$, et donc de cryptanalyser les schémas reposant sur ce problème avec ces tailles de paramètres.

Contributions. Dans [125], avec Guillaume Poupard, nous avons utilisé l'algorithme LLL pour attaquer le schéma de signature RDSA et retrouver la clé privée.

Ce schéma est dérivé, par l'heuristique de Fiat-Shamir [108], du protocole d'authentification zero-knowledge suivant. Soit G un groupe d'ordre inconnu, $\gamma \in G$ et q un nombre premier. La clé secrète est un entier $a \in [2, q-1]$ et la clé publique $\alpha = \gamma^a$. Le prouveur choisit un aléa $k \in [0, q-1]$ et envoie $\rho = \gamma^k$. Le vérifieur choisit un aléa $e \in [0, q-1]$, le challenge, et le prouveur vérifie si $e \in [0, q-1]$. Il calcule ensuite $x = k - a \times e$, $s = x \bmod q$, $\ell = (x-s)/q$ et $\lambda = \gamma^\ell$. Enfin, le vérifieur accepte si $\gamma^s \alpha^e \lambda^q$ est égal à ρ et $s \in [0, q-1]$.

Nous avons montré que le protocole d'authentification n'est pas sûr contre des vérifieurs malhonnêtes choisissant de petits challenges. L'observation principale est que l'aléa k choisi par le prouveur est trop petit pour masquer la clé secrète a dans l'équation $x = k - a \times e$, contrairement au schéma de signature de Schnorr [263]. Le quotient ℓ de x par q est une bonne approximation de $(-a \times e)/q$ car

$$\ell = \left\lfloor \frac{x}{q} \right\rfloor = \left\lfloor \frac{k - a \times e}{q} \right\rfloor = \left\lfloor \frac{-a \times e}{q} \right\rfloor + \varepsilon \text{ où } \varepsilon \in \{0, 1\}. \tag{1.3}$$

La seconde remarque que nous avons utilisée est que la valeur ℓ n'est pas dévoilée, car nous ne connaissons que $\lambda = \gamma^\ell$. Cependant, la taille de ℓ est à peu près celle de e à cause de l'égalité (1.3). Par conséquent, si le challenge e est choisi de petite taille par un vérifieur malhonnête, un algorithme comme la méthode *Rho* de Pollard [246] peut être utilisé pour calculer ℓ.

À partir de ℓ, qui est une bonne approximation de $(-a \times e)/q$, l'adversaire peut en déduire les bits de poids forts de a. En effet, supposons que e est un nombre de taille δ. De

l'équation, $s = k - a \times e - \ell \times q$ avec $0 \leq s < q$, nous obtenons $|a \times e + \ell \times q| < q + |k| < 2q$ et donc,

$$\left| a - \frac{-\ell q}{e} \right| < \frac{2q}{e},$$

et les $(\delta - 1)$ bits de poids forts de a et $-\ell q/e$ sont communs.

Enfin, cette attaque peut être itérée avec des challenges de plus en plus grand ; les bits de poids forts de a peuvent être utilisés pour donner un encadrement du logarithme discret ℓ de λ qui peut être calculé par la méthode des kangorous de Pollard [247].

Dans le cas de la signature, les challenges sont calculés avec une fonction de hachage, $e = H(m\|\rho)$, et ne peuvent donc plus être choisis petits par l'adversaire. Cependant, il est possible de construire des pseudo-signatures en combinant des signatures existantes : pour ces pseudo-signatures, les challenges sont des combinaisons linéaires des challenges initiaux. Les pseudo-signatures sont des entiers qui satisfont les équations de l'algorithme de vérification de signature, mais ne correspondent à aucun message. L'attaque consiste à trouver une combinaison linéaire à petits coefficients donnant un nouveau challenge de petite taille. L'algorithme LLL nous a permis de trouver très efficacement des solutions à ce problème.

Algorithme de Wagner. Quand la densité des instances du problème du sac-à-dos est supérieure à 1 et qu'une seule solution est recherchée, un autre algorithme que LLL peut être utilisé. Cet algorithme repose sur le principe du paradoxe des anniversaires généralisé, et a été redécouvert par Wagner dans [285].

Il permet de résoudre le problème général suivant, appelé k-somme : étant donné $k = 2^q$ listes de m vecteurs aléatoires de n bits, $L_1, \ldots, L_k$, trouver $x_1 \in L_1, \ldots, x_k \in L_k$ tels que $x_1 \oplus \ldots \oplus x_k = 0$.

Une condition nécessaire pour qu'il existe au moins une solution avec une bonne probabilité est que $m^{2^q} \geq 2^n$, c'est-à-dire

$$m \geq 2^{n/2^q}. \tag{1.4}$$

L'algorithme naïf pour résoudre ce problème fonctionne de la manière suivante. Calculer une liste S_1 contenant les sommes $x_1 \oplus \ldots \oplus x_{2^{q-1}}$ et une liste S_2 contenant les sommes $x_{2^{q-1}+1} \oplus \ldots \oplus x_{2^q}$, avec $x_i \in L_i$. Tout vecteur apparaissant à la fois dans S_1 et dans S_2 donne une solution. Il est possible de trouver un tel vecteur en temps linéaire en la taille de S_1 et S_2. Afin d'avoir une probabilité de succès suffisamment grande, nous devons garantir qu'une collision doit exister entre S_1 et S_2. Le paradoxe des anniversaires nous dit qu'il suffit de prendre $|S_1|, |S_2| = \Theta(2^{n/2})$, ce qui donne une complexité en temps en $\tilde{O}(2^{n/2})$. [1]

Dans le cas où la condition (1.4) est une égalité, c'est-à-dire intuitivement lorsque les listes sont courtes, $m = 2^{n/2^q}$, cet algorithme est le plus efficace. Cependant, quand les listes sont plus longues

$$m \geq 2^{n/(q+1)}, \tag{1.5}$$

Wagner a montré dans [285] que le problème peut être résolu en temps $\tilde{O}(2^{q+n/(q+1)})$ en moyenne.

Pour illustrer l'idée principale de cet algorithme, considérons le cas $k = 4$. Soit $L_1, \ldots, L_4$ quatre listes de taille $m = 2^{n/3}$ chacune. Nous procédons en deux étapes. Dans la première étape, nous calculons une liste L'_1 qui contient toutes les sommes $x_1 \oplus x_2$

1. La notation $\tilde{O}$ cache les facteurs logarithmiques dans les complexités en temps, c'est-à-dire ici les facteurs polynomiaux en n, $\log m$ et q.

avec $x_1 \in L_1$ et $x_2 \in L_2$ telles que les $n/3$ premiers bits de la somme soient égaux à zéro. De manière identique, nous calculons une liste L'_2 des sommes des vecteurs de L_3 et L_4 telles que les premiers $n/3$ bits soient égaux à zéro. Alors la taille moyenne de L'_1, et de L'_2, est

$$2^{-n/3} \cdot |L_1| \cdot |L_2| = 2^{n/3}.$$

Dans une deuxième étape, nous trouvons une paire $x'_1 \in L'_1$ et $x'_2 \in L'_2$ telle que $x'_1 \oplus x'_2 = 0$. Comme toute somme d'éléments de L'_1 et L'_2 est nulle sur les $n/3$ premiers bits, la probabilité qu'une somme $x'_1 \oplus x'_2$ aléatoire soit égale à zéro est $2^{-2n/3}$. Par conséquent, le nombre moyen de telles collisions est

$$2^{-2n/3}\mathrm{E}[|L'_1|]\mathrm{E}[|L'_2|] = 2^{-2n/3}2^{n/3}2^{n/3} = 1,$$

et donc nous espérons que l'algorithme trouve une solution. Les listes L'_1 et L'_2 peuvent toutes les deux être calculées en temps $\tilde{O}(2^{n/3})$, comme la collision finale. Cet algorithme a ainsi un temps de calcul moyen de $\tilde{O}(2^{n/3})$, qui est significativement inférieur à $\tilde{O}(2^{n/2})$, la complexité en temps de l'algorithme naïf [2].

Une limitation principale de cet algorithme est qu'il ne fonctionne plus dès que la condition (1.5) n'est plus satisfaite. En effet, dès lors que les listes sont de taille plus petites, c'est-à-dire que $m < 2^{n/(q+1)}$, les listes L' créées n'ont pas d'éléments communs. Minder et Sinclair ont montré dans [218] comment trouver une solution pour toutes les valeurs de m telles que

$$2^{n/2^q} \leq m \leq 2^{n/(q+1)}. \tag{1.6}$$

Par exemple, supposons que nous ayons $m = 2^{2n/7}$ au lieu de $m = 2^{n/3}$ dans notre exemple précédent pour $k = 4$. Si nous éliminons $\ell_1 = n/7$ bits dans la première étape, au lieu de $n/3$ précédemment, alors $\mathrm{E}[|L'_1|] = 2^{-n/7}|L_1||L_2| = 2^{3n/7}$. Nous éliminons maintenant les $\ell_2 = 6n/7$ bits dans la deuxième étape, ce qui donne un nombre moyen de solution égal à

$$2^{-6n/7}\mathrm{E}[|L'_1|]\mathrm{E}[|L'_2|] = 2^{-6n/7}2^{6n/7} = 1;$$

ainsi nous espérons que l'algorithme trouve une solution. Ceci donne un algorithme dont le temps d'exécution moyen est $\tilde{O}(2^{3n/7})$ pour ce choix particulier de paramètres, ce qui est mieux que l'algorithme naïf en $\tilde{O}(2^{n/2})$. Le choix optimal du nombre de bits à éliminer se fait en résolvant un problème d'optimisation écrit sous la forme d'un programme linéaire.

Enfin, quand le nombre de listes est plus grand que la taille binaire des x_i, alors le problème se résout en temps polynomial par de l'algèbre linéaire comme Bellare et Micciancio l'ont montré dans [23]. Supposons par exemple que nous ayons k listes contenant deux éléments de n bits. Il est alors possible d'écrire un système linéaire en utilisant k inconnues y_i booléennes telles que $y_i = 1$ si x_i^1 doit être choisi et $y_i = 0$ sinon. Nous pouvons alors écrire n équations pour chaque bit $j = 1$ à n :

$$\bigoplus_{i=1}^{k} x_i^{1,j} \cdot y_i \oplus x_i^{2,j} \cdot (1 \oplus y_i) = 0$$

Le nombre de contraintes n doit être plus petit que le nombre de degré de liberté k, pour qu'il soit possible de trouver une solution à ce système linéaire. Par conséquent, dès que $k \geq n$, ce système booléen aura une solution si le déterminant du système est non nul. La

2. Les moyennes sont ici prises sur des listes contenant des entrées aléatoires. Les algorithmes sont déterministes.

probabilité qu'un système aléatoire booléen soit inversible est supérieure à 1/2 à partir de $k \geq n + 1$.

Remarques : Bien avant l'article de Wagner en 2002, cet algorithme a été utilisé par Patarin et Camion dès 1991 dans [59] pour inverser la fonction de hachage proposée par Damgård dans [85] et basée sur le problème du sac-à-dos. De même, Joux et Lercier l'avaient utilisé pour calculer efficacement une étape dans le calcul du logarithme discret dans [166]. Ultérieurement Chose, Joux, et Mitton dans [70] l'ont appliqué pour rechercher des équations de parité dans les chiffrements par flot utilisant des registres à décalages linéaires. Récemment, il a été utilisé pour résoudre les instances du sac-à-dos quand $d \leq 1$. Bien que dans ce cas la solution soit unique, Howgrave-Graham et Joux [156] ont remarqué que cette solution peut s'écrire avec plusieurs représentations et nous pouvons rechercher une représentation particulière de la solution. Pour ces instances, la complexité de leur algorithme est en $\tilde{O}(2^{0.385n})$ en temps et $\tilde{O}(2^{0.247n})$ en mémoire.

Contributions. Avec Jacques Stern et Sébastien Zimmer [127], nous avons utilisé cet algorithme pour trouver des collisions sur la fonction de hachage Smash proposée par Knudsen [190]. L'originalité de la fonction de compression de Smash est d'utiliser un appel à une permutation aléatoire, π, instanciée par un schéma de chiffrement par bloc sous une clé fixe :

$$smash^{\pi}(h, x) = \pi(h \oplus x) \oplus h \oplus \theta \cdot x,$$

où $\cdot$ désigne la multiplication dans $\mathbb{F}_{2^n}$ et $\theta \neq 0, 1$, un élément fixe de $\mathbb{F}_{2^n}$. La fonction de hachage itère cette fonction de compression en utilisant le mode de Merkle [216] et Damgård [85]. Cette fonction a été renforcée, suite à une attaque de Pramstaller, Rechberger et Rijmen dans [248], en utilisant une permutation différente à chaque itération.

Nous allons chercher des collisions sur des messages de deux blocs. Les indices 1 et 2 représentent les deux messages qui collisionnent et les primes à la seconde itération. Soit π et π', deux permutations utilisées pendant ces itérations et (α_1, β_1) et (α'_1, β'_1) deux couples tels que $\pi(\alpha_1) = \beta_1$ et $\pi'(\alpha'_1) = \beta'_1$. Définissons une valeur intermédiaire pour calculer les sorties de smash sur deux itérations : $\gamma_1 = \beta_1 \oplus \theta \cdot \alpha_1$, $\gamma'_1 = \beta'_1 \oplus \theta \cdot \alpha'_1$. Le message d'entrée $x_1 = \alpha_1 \oplus h_0$ a pour sortie, $h_1^1 = smash^{\pi}(h_0, x_1) = \gamma_1 \oplus (\theta + 1) \cdot h_0$. Pour la seconde itération, le message $x'_1 = \alpha'_1 \oplus h_1^1$ a pour image, $h_2^1 = smash^{\pi'}(h_1^1, x'_1)$. Nous pouvons alors exprimer le calcul de h_2^1 en fonction de h_0, et des deux valeurs intermédiaires γ_1, γ'_1 :

$$h_2^1 = \gamma'_1 \oplus (\theta + 1) \cdot \gamma_1 \oplus (\theta + 1)^2 \cdot h_0.$$

De même, nous avons la relation pour le deuxième message :

$$h_2^2 = \gamma'_2 \oplus (\theta + 1) \cdot \gamma_2 \oplus (\theta + 1)^2 \cdot h_0.$$

Si nous voulons une collision à la fin de la seconde itération, $h_2^1 = h_2^2$, nous obtenons

$$(\theta + 1) \cdot \gamma_1 \oplus (\theta + 1) \cdot \gamma_2 \oplus \gamma'_1 \oplus \gamma'_2 = 0.$$

Dans cette relation, $\gamma_1, \gamma'_1, \gamma_2, \gamma'_2$ sont des valeurs aléatoires. En utilisant l'algorithme de Wagner, nous voyons que dès que la taille des listes $L_{\gamma_1}, L_{\gamma'_1}, L_{\gamma_2}$ et $L_{\gamma'_2}$ contenant les valeurs $\gamma_1, \gamma'_1, \gamma_2, \gamma'_2$ dépassent $2^{n/3}$, nous pouvons trouver une collision en temps et mémoire $\tilde{O}(2^{n/3})$. Cette attaque peut être étendue en utilisant plus de blocs et au final, la complexité est $\tilde{O}(2^{2\sqrt{n}})$ pour des messages de $2^{\sqrt{n}-1}$ blocs.

Nous avons également proposé dans [127] une réparation de cette fonction de compression avec une preuve de sécurité en remplaçant la multiplication par θ par une autre

permutation aléatoire. Cette réparation a été utilisée pour construire la fonction Grøstl, candidate à la compétition SHA-3.

Application de l'algorithme de Wagner. L'algorithme de Wagner a été utilisé pour trouver des collisions dans une fonction de hachage dont la sécurité repose sur un problème difficile de la théorie des codes correcteurs. Cette fonction de hachage, appelée *Fast Syndrome Based* (FSB), a été proposée par Augot, Finiasz et Sendrier dans [11] et repose sur le problème difficile de trouver dans un code linéaire aléatoire un mot de code ayant un syndrome donné. Ce problème a été prouvé NP-difficile par les concepteurs de FSB dans [11].

Les messages sont transformés en mots du code et leur image par la fonction est le syndrome de ce mot de code. Pour calculer le haché $h(M)$ de M, il suffit d'encoder M en un mot de code $\varphi(M)$ par une fonction injective et d'appliquer la matrice de parité $\mathcal{H}$ de taille $r \times n$:

$$h(M) = \mathcal{H} \times \varphi(M).$$

Une fonction d'encodage a également été proposée. Elle transforme un message M de s bits en un mot de code $\varphi(M)$ de n bits ayant un poids de Hamming égal à w de telle sorte que le mot de code a exactement un bit à 1 par bloc de n/w bits. Elle consiste à découper M en bloc M_i de $\log(n/w)$ bits et à appliquer la fonction ψ de $\{0,1\}^{\log(n/w)}$ vers $\{0,1\}^{n/w}$ qui associe à x sa position dans les n/w valeurs possibles. Le calcul de h s'écrit alors :

$$h(M) = \mathcal{H} \times \varphi(M) = \bigoplus_{i=0}^{w-1} \mathcal{H}_{in/w+\psi(M_i)},$$

où $\mathcal{H}_i$ représente la $i$ème colonne de la matrice $\mathcal{H}$. La fonction de hachage consiste à sommer w colonnes de taille r de la matrice $\mathcal{H}$. Chacune de ces colonnes est choisie dans un ensemble de n/w colonnes.

Pour calculer une collision sur cette fonction de hachage, il suffit de trouver M et M' tels que

$$h(M) \oplus h(M') = \bigoplus_{i=0}^{w-1} \mathcal{H}_{in/w+\psi(M_i)} \oplus \bigoplus_{i=0}^{w-1} \mathcal{H}_{in/w+\psi(M'_i)} = 0.$$

Il s'agit d'une instance du problème résoluble par l'algorithme de Wagner avec $2w$ listes de vecteurs de r bits où la taille des listes est n/w comme Coron et Joux l'ont montré dans [77].

Les paramètres de FSB ont alors été modifiés par Finiasz, Gaborit et Sendrier en 2007 pour éviter cette attaque et ces derniers ont également proposé dans [109] de réduire la taille mémoire de la matrice en choisissant une matrice $\mathcal{H}$ cyclique par bloc :

$$\mathcal{H} = \mathcal{H}_0 \| \mathcal{H}_1 \| \dots \mathcal{H}_{w-1},$$

où chaque $\mathcal{H}_i$ est une matrice $r \times r$ cylique. La nouvelle fonction de hachage s'appelle *Improved FSB* (IFSB). Malheureusement, certains des nouveaux paramètres de cette fonction sont faibles quand $2w \geq r$. En effet, il s'agit alors d'une instance du problème k-somme avec un nombre de listes $2w$ plus grand que la taille des vecteurs r qui se résout donc par algèbre linéaire.

Contributions. Avec Gaëtan Leurent nous avons étendu cette attaque en collision contre IFSB dans [116] en prenant en compte la structure cyclique par bloc de la matrice.

Introduisons quelques notations : une chaîne binaire x est périodique de longueur ℓ si $x = x \lll \ell$, où $x \lll \ell$ représente la rotation de x de ℓ bits vers la gauche. De manière

identique, si x est coupé en bloc de longueur k, $x = x_0 \| x_1 \| x_2 \ldots$, alors x est périodique par bloc si $x \overset{k}{\lll} \ell = x$, où

$$x \overset{k}{\lll} \ell = (x_0 \lll \ell) \| (x_1 \lll \ell) \| (x_2 \lll \ell) \| \ldots .$$

Si $\mathcal{H}_i$ est cyclique, alors $\mathcal{H}_i \times (x \lll \ell) = (\mathcal{H}_i \times x) \lll \ell$ et dans le cas où $\mathcal{H} = \mathcal{H}_0 \| \mathcal{H}_1 \| \mathcal{H}_2 \ldots$ est quasi-cyclique,

$$\mathcal{H} \times (x \overset{k}{\lll} \ell) = \bigoplus_{i=0}^{w-1} \mathcal{H}_i \times (x_i \lll \ell) = (\mathcal{H} \times x) \lll \ell.$$

Nous avons montré que si nous prenons un message M tel que $\varphi(M)$ est périodique par bloc, alors son image par la fonction de compression est périodique. Par conséquent, nous pouvons maintenant rechercher des collisions sur la longueur d'une période et nous aurons alors une collision sur toute la fonction. Cette étape est alors une étape préliminaire à l'algorithme de Wagner ou à l'attaque par algèbre linéaire.

1.2 Systèmes polynomiaux

À côté des problèmes du sac-à-dos et de ceux issus de la théorie des codes correcteurs d'erreurs qui s'expriment sous la forme de systèmes linéaires contraints, le cryptanalyste doit résoudre divers problèmes algébriques autour des polynômes. Ces problèmes consistent par exemple à trouver de petites racines modulaires d'un polynôme ou les petites racines entières d'un polynôme à plusieurs variables.

1.2.1 Petites racines d'un polynôme à une variable

Dans [75], Coppersmith a proposé un algorithme utilisant l'algorithme LLL pour trouver les petites racines d'une équation polynomiale en une variable modulo un module RSA. Cet algorithme propose une réduction entre le problème consistant à trouver les petites racines modulaires d'un polynôme univarié et le problème plus facile de résoudre une équation polynomial en une variable dans $\mathbb{Z}$. Il utilise la réduction de réseau pour trouver une équation polynomiale entière satisfaite par toutes les petites racines de l'équation initiale $P(x) = 0 \bmod N$. La preuve a été simplifiée dans [155].

Premier théorème de Coppersmith. *Soit $P(x)$ un polynôme unitaire de degré δ à coefficients entiers et N un entier de factorisation inconnue. Nous pouvons alors trouver en temps polynomial en $(\log N, \delta)$ tous les entiers x_0 tels que $P(x_0) = 0 \bmod N$ et $|x_0| \leq N^{1/\delta}$.*

Une application directe de ce théorème est l'inversion de la fonction RSA sur de petits messages. Soit N un module RSA de n bits et m le message à chiffrer, que nous supposons "petit", c'est-à-dire par exemple de 128 bits si m est une clé de session symétrique à transmettre. Supposons par ailleurs que l'exposant public soit $e = 3$, choix fréquent permettant d'accélérer les calculs lors du chiffrement. Alors pour un module de 1024, l'inégalité $|m| \leq N^{1/e}$ est vérifiée. Nous pouvons remarquer que si le chiffré se calcule par $c = m^e \bmod N$, alors $m^e < N$ et $c = m^e \in \mathbb{Z}$. Pour retrouver m à partir de c, il suffit de calculer une racine e-ième d'un polynôme, ce qui peut être réalisé par l'algorithme de Newton.

Si le schéma de chiffrement RSA utilise tout d'abord une fonction d'encodage déterministe et connue, nous pouvons voir que l'entrée de la fonction RSA s'écrit alors $a \times 2^{n/e} + m$

pour a un entier connu. La fonction d'encodage ajoute à la chaîne binaire représentant l'entier m, une chaîne binaire fixe représentant l'entier a. Le message m chiffré avec RSA est alors l'entier $(a \times 2^{n/e} + m)^e \bmod N$. Pour retrouver le message m à partir de son chiffré c, le cryptanalyste doit donc trouver les petites racines du polynôme $P(x) = (a \times 2^{n/e} + x)^e - c$, ce qui peut se faire en temps polynomial d'après le théorème précédent.

Coppersmith a également montré dans [75] un second théorème très utile pour factoriser un entier si un de ses facteurs s'obtient comme l'évaluation d'un polynôme connu en une petite valeur. Nous présentons ici une version un peu différente dûe à Boneh, Durfee et Howgrave-Graham [48].

Deuxième théorème de Coppersmith. *Soit $P(x) \in \mathbb{Z}[x]$ un polynôme unitaire de degré δ en une variable et soit N en entier de factorisation inconnue. Soit $\alpha \in \mathbb{Q}$ tel que $0 \leq \alpha \leq 1$. Nous pouvons alors trouver en temps polynomial en $(\log N, \delta)$ et la taille de α, tous les entiers x_0 tels que $\operatorname{pgcd}(P(x_0), N) \geq N^\alpha$ et $|x_0| \leq N^{\alpha^2/\delta}$.*

L'application la plus célèbre de ce second théorème est la factorisation d'un module $N = pq$ si par exemple, la moitié des bits de poids fort de p est connue. En effet, écrivons p de la forme $p = 2^{n/4} p_0 + p_1$ et supposons que p et q sont de même taille avec $p < q$. Soit p_0 les bits de p connus, $p_1 \leq N^{1/4}$, et considérons le polynôme $P(x) = 2^{n/4} p_0 + x$ et $\alpha = 1/2$. Comme $\deg P = 1$ et $\alpha^2/\delta = 1/4$, nous pouvons alors retrouver p_1 en temps polynomial.

Autre application : Retrouver d à partir de la connaissance de certains bits. Dans les attaques par canaux auxiliaires, il est parfois possible d'obtenir par des mesures physiques des bits des éléments secrets, consécutifs ou non. Le cryptanalyste doit alors retrouver les clés entièrement à partir de ces informations partielles.

Considérons la fonction RSA avec un exposant public petit. Lorsque l'exposant public e est petit, la moitié des bits de poids forts de d est connue. En effet, pour p et q de même taille, c'est-à-dire $p < \sqrt{N} < q < 2p$, alors $|N - \varphi(N)| = |p + q - 1| \leq 3 \cdot N^{1/2}$, ce qui signifie que N est une bonne approximation de $\varphi(N)$ car environ la moitié des bits de poids fort est commune. En remplaçant $\varphi(N)$ par $N + 1$ dans l'équation $ed - k\varphi(N) = 1$, nous avons :

$$d_{msb} = \left\lfloor \frac{1 + k(N+1)}{e} \right\rfloor.$$

La distance entre d et d_{msb} peut être majorée :

$$|e(d - d_{msb})| = |k| \cdot |N - \varphi(N)| < 3|k|N^{1/2}.$$

D'après l'algorithme d'Euclide étendu, nous avons $d < \varphi(N)$ et l'inégalité $k\varphi(N) = ed - 1 < ed$ nous permet de déduire que $k < e$. Par conséquent, $|d - d_{msb}| < 3N^{1/2}$ et si e est petit, une recherche exhaustive sur k permet de calculer d_{msb}.

Boneh, Durfee et Frankel ont montré dans [47] comment utiliser le deuxième théorème de Coppersmith pour retrouver d quand par exemple les $n/4$ bits de d en position $n/4$ à $n/2$ sont connus.

Notons d_1 les bits connus de d et écrivons $d = 2^{n/4} d_1 + d_0$ avec $|d_0| < 2^{n/4}$. Remarquons que les bits de d, en position $n/2$ à n, qui correspondent aux $n/2$ bits de poids fort de d_1, ont a priori e valeurs possibles. Une recherche exhaustive de k de, 1 à e, va permettre de retrouver d_0 et les bits de poids fort de d_1. Définissons

$$s_1 = 1 + N - \frac{ed_1 - 1}{k'}.$$

Pour $k' = k$, nous obtenons, pour $s = p + q$,

$$|s_1 - s| = \left|\frac{e}{k}(d - d_1)\right| < 2^{(n/4)+\log_2 e}.$$

Si $p > q$, $p = \frac{1}{2}(s + \sqrt{s^2 - 4N})$. Posons alors

$$p_1 = \frac{1}{2}(s_1 + \sqrt{s_1^2 - 4N}).$$

Il est facile de vérifier que p_1 est une bonne approximation de p,

$$|p - p_1| \leq 2^{(n/4)+5+\log_2 e}.$$

Par conséquent, en utilisant une recherche exhaustive de $32e$ étapes, nous pouvons retrouver la moitié des bits de poids fort de p et le second théorème de Coppersmith permet alors de retrouver p en entier.

Les attaques physiques permettent d'obtenir des bits de la clé privée d ou des facteurs p et q de N. Nous décrirons dans le chapitre sur les méthodes physiques, certaines attaques de ce type qui retrouvent la clé d bit par bit en commençant par les poids faibles. Dans ce cas, les algorithmes utilisant LLL comme ceux de Boneh, Durfee et Frankel permettent alors d'éviter d'avoir à retrouver complètement la clé car un quart des bits suffisent. Dans d'autres cas, il faut concevoir de nouveaux algorithmes car les bits des éléments secrets obtenus ne sont pas *contigus* et en général, l'algorithme LLL n'est pas adapté dans ces situations.

1.2.2 Retrouver d à partir de bits non-consécutifs des éléments secrets

Les attaques physiques par *congélation de la mémoire* permettent d'obtenir certains bits des éléments secrets. Les éléments permettant de déchiffrer sont chiffrés sur le disque dur sous un mot de passe dans les logiciels de sécurité. L'utilisateur légitime s'authentifie au lancement du programme et les éléments secrets sont stockés dans la mémoire vive pendant toute l'utilisation du logiciel. Un attaquant qui vole un portable en cours d'utilisation ou un serveur SSL, peut utiliser une bombe de froid pour congeler la mémoire vive. La rémanence de la mémoire a pour effet que la mémoire conserve une partie des informations. Par conséquent, s'il est possible de savoir à quelles adresses sont stockées les éléments secrets, l'attaquant peut obtenir certains bits secrets.

Dans le cas d'un déchiffrement RSA utilisant le théorème des restes chinois, les éléments p, q, $d_p = d \bmod (p-1)$, $d_q = d \bmod (q-1)$ stockés en mémoire vive. À partir des bits non consécutifs connus de p, q, d, d_p et d_q, nous pouvons factoriser le module N en écrivant un système correspondant aux équations binaires de

$$\begin{aligned} N &= pq && (1.7)\\ ed &= k(N - p - q + 1) + 1 && (1.8)\\ ed_p &= k_p(p - 1) + 1 && (1.9)\\ ed_q &= k_q(q - 1) + 1 && (1.10) \end{aligned}$$

et aux bits connus. L'algorithme de reconstruction de Heninger et Shacham dans [151] énumère toutes les clés partielles possibles et élimine celles qui ne satisfont pas les contraintes des bits connus. Plus précisément, étant donné les bits de la position 1 à $i - 1$ d'une clé potentielle, générer toutes les combinaisons pour le bit i de p, q, d, d_p, d_q, et conserver les

combinaisons qui satisfont les contraintes (1.7), (1.8), (1.9) et (1.10) mod 2^i. Il est aussi possible de prendre en compte des erreurs dans les mesures comme l'ont fait Hanecka, May et Meurer dans [150].

Contributions. Ces algorithmes ne permettent pas de retrouver les bits manquant de d si nous connaissons seulement des bits de d.

Dans [114], avec Sébastien Kunz-Jacques, Gwenaëlle Martinet, Frédéric Muller et Frédéric Valette, nous avons développé un algorithme pour ce cas si nous sommes capable d'obtenir des bits non-consécutifs de versions randomisées de d, c'est-à-dire de la forme $d+\lambda\varphi(N)$. Cette hypothèse est raisonnable car de nombreuses implémentations sécurisées utilisent cette randomisation pour contrer certaines attaques physiques. La valeur de λ est une valeur aléatoire de petite taille 20 ou 32 bits afin de ne pas trop pénaliser l'algorithme d'exponentiation dont la complexité est linéaire en le nombre de bits de l'exposant. Nous supposons qu'à chaque itération nous pouvons obtenir une fraction $1/r$ des bits de $d+\lambda\varphi(N)$. Cette hypothèse est justifiée dans le chapitre sur les attaques physiques.

Enfin, plaçons-nous dans le cas d'exposant public de petite taille, ce qui est le cas en pratique où $e=3$ ou $e=17$. Dans ce cas, la moitié des bits de poids fort de d est connue ainsi que la valeur de k associée, telle que $ed=1+k\varphi(N)$. Nous avons vu qu'une recherche exhaustive permet de les obtenir quand e est petit. Dans le cas $e=3$, il est également possible de montrer que $ed=1+2\varphi(N)$ [114].

Notre algorithme fonctionne en deux étapes. Dans une première étape, nous recherchons les valeurs de λ associées à chaque exponentiation. Pour ce faire, nous regardons les bits de poids forts de $d+\lambda\varphi(N)$. Comme la moitié des bits de poids forts de d et 40% des bits de $d+\lambda\varphi(N)$ sont connus, nous pouvons alors écrire des équations binaires entre ces bits. Si le nombre de contraintes $(n/2r)$ est supérieur au nombre de degrés de liberté 20 ou 32, alors en essayant toutes les valeurs de λ, nous pouvons déterminer la bonne valeur de λ avec forte probabilité. Nous avons donc pour chaque valeur $d+\lambda\varphi(N)$, la valeur de λ qui a été utilisée.

La deuxième étape permet de retrouver les octets de $\varphi(N)$ en commençant par les octets de poids faibles ainsi que ceux de d. Notons que nous avons besoin de retrouver uniquement la première moitié des bits de $\varphi(N)$. Nous allons exploiter l'équation,

$$d \bmod 2^8 = \frac{1+k\varphi(N)}{e} \bmod 2^8, \tag{1.11}$$

en faisant une recherche recherche exhaustive sur les valeurs de $\varphi(N) \bmod 2^8$. L'équation (1.11) est valide car e est bien inversible modulo 2^8. Nous avons alors l'équation

$$d_{observé} = \frac{1+k\varphi(N)}{e} + \lambda\varphi(N) \bmod 2^8,$$

où le membre de droite est parfaitement connu. Si nous avons m mesures de $d_{observé} \bmod 2^8$, qui nous apporte 8 bits de contraintes, dès que $8 \times m/r \geq 8$, nous pouvons retrouver $\varphi(N) \bmod 2^8$. Il suffit ensuite de recommencer avec l'octet suivant de $\varphi(N)$. Dans certains cas, il nous reste plusieurs choix pour les octets de $\varphi(N)$. Nous nous sommes aperçus qu'à l'étape suivante, les mauvaises s étaient rapidement éliminées.

Notre algorithme est capable de retrouver la clé secrète pour des modules de 1024 bits dès que $1/r = 1/16$. Les détails techniques de ce résultat ainsi que l'extension pour des exposants plus grands par exemple, $e=2^{16}+1$, sont proposés dans l'annexe J. Dans ce cas, nous avons optimisé l'attaque pour que la complexité reste pratique car la recherche exhaustive sur k et sur λ devient coûteuse.

1.2.3 Approximation diophantienne simultanée

L'approximation diophantienne simultanée a été utilisée la première fois en cryptographie par Lagarias pour attaquer un système de chiffrement basé sur le problème du sac-à-dos dans [199]. Elle est aussi utilisée dans un algorithme de factorisation proposé par Schnorr dans [264].

L'approximation diophantienne étudie l'approximation de nombres réels par des rationnels. Dans le cas d'un seul réel, le problème est assez simple car nous pouvons approcher tout réel par des rationnels quelle que soit la précision voulue. Par exemple, Dirichlet montra le théorème suivant en utilisant le lemme des tiroirs :

Théorème de Dirichlet. *Soit $x \in \mathbb{R}$. Il existe une infinité de $q \in \mathbb{Z}$ tels que*

$$\left| x - \frac{p}{q} \right| < \frac{1}{|q|^2},$$

pour un certain $p \in \mathbb{Z}$.

Le problème de l'approximation d'un vecteur de réels par un vecteur à coordonnées rationnelles, de même dénominateur, est plus difficile car nous perdons en précision. Le théorème de Minkowski [62] permet de montrer le résultat suivant :

Théorème. *Pour tout $y \in \mathbb{R}^n$, il existe une infinité de $q \in \mathbb{Z} \setminus \{0\}$ tels que*

$$\|qy - p\|^n \cdot |q| < 1,$$

pour un certain $p \in \mathbb{Z}^n$.
Si y_i désigne la $i$ème composante de y et p_i celle de p, nous avons :

$$|qy_i - p_i| \leq \|qy - p\|,$$

et donc,

$$|qy_i - p_i|^n \cdot |q| < 1, \text{ soit } \left| y_i - \frac{p_i}{q} \right| < \frac{1}{|q|^{1+\frac{1}{n}}}.$$

Nous pouvons nous demander s'il existe des vecteurs ayant de "bonnes" ou de "mauvaises" approximations. Schmidt dans [262] définit les notions de vecteur très bien approchable (TBA) et très mal approchable (TMA).

Définition :

1. Un vecteur $y \in \mathbb{R}^n$ est dit TMA si et seulement s'il existe $c > 0$ tel que

   $$\|yq - p\|^n \cdot |q| \geq c,$$

   pour tout $p \in \mathbb{Z}^n$ et tout $q \in \mathbb{Z} \setminus \{0\}$.
2. Un vecteur $y \in \mathbb{R}^n$ est dit TBA si et seulement s'il existe $\varepsilon > 0$ pour lequel il y a une infinité de $q \in \mathbb{Z} \setminus \{0\}$ tels que

   $$\|yq - p\|^n \cdot |q| \leq |q|^{-\varepsilon},$$

   pour un certain $p \in \mathbb{Z}^n$.

En munissant l'espace $\mathbb{R}^n$ de la mesure de Lebesgue, nous pouvons montrer que presque tout vecteur de $\mathbb{R}^n$ n'est pas TBA.

Considérons le problème algorithmique de trouver une bonne approximation dans le cas où le vecteur y est un vecteur de rationnels. Soit $\left(\frac{p_1}{p}, \frac{p_2}{p}, \ldots, \frac{p_n}{p}\right)$ un vecteur de rationnels, et δ un nombre réel. Trouver une approximation diophantienne simultanée de qualité δ, consiste à trouver un vecteur $\left(\frac{q_1}{q}, \frac{q_2}{q}, \ldots, \frac{q_n}{q},\right)$ de rationnels tel que $q < p$ et

$$\left|\frac{p_i}{p}q - q_i\right| \leq p^{-\delta}, \text{ pour } i = 1, 2, \ldots, n.$$

L'approximation est dite *anormalement bonne* quand $\delta > \frac{1}{n}$.

Lagarias a proposé dans [200] un algorithme, appelé SDA pour réduire le problème de trouver une approximation anormalement bonne au problème de trouver un vecteur court dans un réseau. Considérons en effet le réseau engendré par les lignes de la matrice suivante avec $K \approx p^\delta$:

$$\begin{pmatrix} Kp & 0 & 0 & \ldots & 0 & 0 \\ 0 & Kp & 0 & \ldots & 0 & 0 \\ 0 & 0 & Kp & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & Kp & 0 \\ -Kp_1 & -Kp_2 & -Kp_3 & \ldots & -Kp_n & 1 \end{pmatrix}.$$

En multipliant cette matrice à gauche par le vecteur $(q_1, q_2, \ldots, q_n, q)$, on obtient le vecteur cible du réseau de coordonnées $(K(q_1p - qp_1), K(q_2p - qp_2), \ldots, K(q_np - qp_n), q)$ qui est de norme $\sqrt{n+1}p$. Ce vecteur est petit comparé aux vecteurs d'origine de la matrice dont la taille est environ $p^{1+\delta}$.

Contributions. Dans [121], avec Gwenaëlle Martinet et Guillaume Poupard, nous avons utilisé l'algorithme SDA pour résoudre le problème suivant : étant donné $N = pq$ et des multiples bruités de p, entiers de la forme $s_i = r_i + u_ip$ avec $|r_i| \ll p$, factoriser N. Nous avons rencontré ce problème dans une attaque physique sur l'implémentation de l'algorithme de Garner pour calculer RSA avec le théorème des restes chinois. Le temps de calcul de l'algorithme permet de savoir si dans une signature RSA s, que nous pouvons écrire de la forme $s = r + up$, le r est petit ou non. Nous pouvons alors collecter plusieurs signatures de cette forme et nous pouvons nous demander quelle doit être la taille de r pour qu'il soit possible de factoriser N.

Notre algorithme s'exécute en temps polynomial en le nombre de signatures avec probabilité supérieure à $1 - \varepsilon$, en ayant m multiples bruités de p tels que $|r_i| < p/2^\ell$, dès que

$$\ell > \max\left(\frac{\log q - \log \varepsilon - 1.105}{m} + 2.047, \log(2\sqrt{m+1})\right)$$

si LLL retourne vers le vecteur le plus court. Par exemple, pour $m = 60$ et un module de 1024 bits, nous retrouvons le facteur q, avec probabilité supérieure à $1 - 2^{-10}$ dès que $\ell \geq 11$.

Nous avons montré que ce problème se réduit au problème de trouver une approximation simultanée anormalement bonne du vecteur $(\frac{s_1}{N}, \ldots, \frac{s_m}{N})$. Une telle approximation dans notre cas existe et elle est donnée par le vecteur

$$\left(\frac{u_1}{q}, \ldots, \frac{u_m}{q}\right).$$

En effet, pour $i = 1, \ldots, m$, nous avons l'équation :

$$\left| q\frac{s_i}{N} - u_i \right| = |r_i|/|p| \leq 2^{-\ell} \leq N^{-\delta},$$

avec $\delta > 1/m$, nous en déduisons que $\ell \geq (\log N)/m$. Notre analyse dans [121] montre que le bruit ℓ dépend de $(\log q)/m$ et non de $(\log N)/m$. Les détails de ce résultat sont proposés dans l'annexe I.

Contributions. Nous avons également donné une autre application de ce problème dans [111] avec Nick Howgrave-Graham pour factoriser un module $N = p^2q$ dans le cas des signatures ESIGN [129].

La sécurité de ce schéma de signature est basée sur le problème de trouver une approximation d'une racine e-ième modulo N à un facteur additif $2(\log N)/3$ près. Dans ce schéma, le module N est de la forme p^2q avec p et q de même taille k. La signature du message $m \in \mathbb{Z}_N$ consiste à trouver un entier s tel que $|s^e \bmod N - m| \leq 2^{2k}$. Pour trouver un tel entier s, l'algorithme choisit r aléatoirement dans $\mathbb{Z}_{pq}$ et en déduit $u \bmod p$ tel que $s = r + upq$. Pour déterminer u, le signataire calcule la valeur $v = \lceil ((m - r^e) \bmod N)/(pq) \rceil$ et pose $u = v \cdot (er^{e-1})^{-1} \bmod p$. Cet algorithme de signature est plus rapide que RSA car il nécessite une exponentiation $\bmod N$ et une $\bmod p$ de petite taille et une inversion $\bmod p$. En général, la valeur de e est 4 ou 8.

Nous avons montré que s'il est possible d'obtenir les bits de poids forts des aléas r, alors nous pouvons obtenir à partir des signatures des entiers de la forme $s = r + upq$, avec r petit. Il est possible de réduire ce problème à l'algorithme SDA pour factoriser N en temps polynomial.

1.2.4 Système d'équations quadratiques à plusieurs variables

La cryptographie multivariée est une branche de la cryptographie à clé publique dans laquelle la sécurité des schémas est basée sur le problème NP-difficile [134] de résoudre un système d'équations quadratiques en plusieurs variables. Dans ces systèmes, la clé publique est composée d'un système P d'équations quadratiques en plusieurs variables. Le chiffrement du message m consiste simplement à évaluer le système, $P(m) = c$, et le déchiffrement à l'inverser, $P^{-1}(c)$. Dans les schémas de signature, le système peut avoir plusieurs inverses. L'algorithme de signature calcule un antécédent s au message m et l'algorithme de vérification teste si s est bien un inverse de m, c'est-à-dire $P(s) = m$, en calculant $P(s)$.

Pour inverser efficacement la clé publique P, les concepteurs des systèmes utilisent des systèmes quadratiques facilement inversibles f et les masquent grâce à des changements secrets et linéaires de variables sur les entrées, S, et les sorties, T :

$$P = T \circ f \circ S.$$

Pour retrouver les éléments secrets S et T ou seulement inverser ces systèmes, le cryptanalyste étudie les *formes bilinéaires symétriques* associées aux formes quadratiques du système considéré. L'étude de ces formes bilinéaires, comme par exemple la recherche de leurs applications auto-adjointes, permet de retrouver les éléments secrets.

Schéma de chiffrement C^* de Matsumoto et Imai. Le système de chiffrement C^* de Matsumoto et Imai [212] a été un des premiers schémas proposés en cryptographie multivariée et apparaît comme le père de toute une famille de schémas de chiffrement, HFE [240] et PMI [90], et de signature, SFLASH [241] et C^{*-} projeté [91], ...

Le schéma C^* utilise la fonction monomiale $f(X) = X^{1+q^\theta}$ dans le corps fini $\mathbb{F}_{q^n}$. Cette fonction monomiale est facilement inversible car dans un corps fini, l'ordre du groupe multiplicatif est connu, égal à $q^n - 1$. Ceci permet de calculer l'inverse de f, $f^{-1}(X) = X^\delta$, avec $\delta = (1 + q^\theta)^{-1} \bmod (q^n - 1)$. Le système d'équations quadratiques en n variables est obtenu en projetant l'équation

$$B = A^{1+q^\theta} \text{ dans } \mathbb{F}_{q^n},$$

sur une base de $(\mathbb{F}_q)^n$. Ces équations sont quadratiques car $f(X) = X \cdot X^{q^\theta}$ et la fonction Frobenius $X \mapsto X^{q^\theta}$ est une application linéaire de $(\mathbb{F}_q)^n$ dans $(\mathbb{F}_q)^n$. La clé publique est constituée des équations de $P = T \circ f \circ S$, et la clé secrète des matrices S et T qui permettent d'inverser le système puisqu'elles sont inversibles. Comme les changements de variables linéaires S et T ne modifient pas le degré des équations, les équations de la clé publique sont également quadratiques.

La difficulté de retrouver la clé secrète S et T à partir de la clé publique $P = T \circ f \circ S$ ne provient pas de la matrice T, car nous pouvons toujours écrire l'équation de la clé publique sous la forme $T^{-1} \circ P = f \circ S$. Nous obtenons un système d'équation linéaire en les n^2 inconnues de T^{-1} et quadratique en les n^2 inconnues de S. Les algorithmes génériques pour résoudre de tels systèmes, comme les algorithmes calculant des bases de Gröbner [105], ne sont pas efficace quand le système quadratique induit par f est homogène, c'est-à-dire qu'il ne contient que des monômes de même degré comme l'ont montré expérimentalement Faugère et Perret dans [104]. Ces algorithmes sont capables de retrouver la matrice S quand le nombre de variables quadratiques est petit $n \leq 20$. Enfin, quand des équations sont enlevées à la clé publique, ces algorithmes sont rapidement inefficaces car le nombre de solutions augmentent.

Existence de relations bilinéaires. Dans [238], Patarin a montré qu'il est possible d'inverser la fonction de chiffrement de C^* sans avoir besoin de retrouver la clé secrète. Cette attaque repose sur l'existence et le calcul efficace de relations *bilinéaires* $B(X, Y)$ entre les entrées X et sorties Y du système P. Une fois que ces relations sont obtenues, elles permettent de déchiffrer efficacement n'importe quel chiffré c. En effet, en remplaçant Y par c dans ces relations, nous obtenons des relations *linéaires* sur le message X. Si l'adversaire obtient n relations linéaires *indépendantes*, il peut alors reconstruire le message en inversant le système linéaire. Les équations bilinéaires représentent une clé secrète équivalente à la clé privée S, T. L'existence de relations bilinéaires provient de la forme monomiale de la fonction f.

Nous allons montrer qu'il existe n relations bilinéaires entre le message et le chiffré. Regardons dans un premier temps le système $f : b = f(a) = a^{1+q^\theta}$, avec $a, b \in \mathbb{F}_{q^n}$. Nous pouvons voir que l'élévation de cette équation à la puissance $q^\theta - 1$, puis la multiplication pas ab, permet d'obtenir la relation suivante :

$$ab^{q^\theta} = a^{q^{2\theta}} b. \tag{1.12}$$

Compte tenu que les élévations à une puissance de q sont des applications linéaires sur $(\mathbb{F}_q)^n$, en projetant l'équation (1.12) sur une base de $(\mathbb{F}_q)^n$, nous obtenons n relations bilinéaires entre les entrées $a = (a_0, \ldots, a_{n-1})$ et sorties $b = (b_0, \ldots, b_{n-1})$. Enfin, les applications linéaires S et T ne changent rien au caractère bilinéaire de ces équations. Il existe donc des relations bilinéaires, dépendant de S et T, entre les entrées et sorties de la clé publique P. Il y a bien n relations bilinéaires *indépendantes* car le système P est inversible.

L'attaque consiste donc dans un premier temps à rechercher ces relations. Nous pouvons les écrire sous la forme

$$\sum_{0\leq i,j\leq n-1} \alpha_{i,j} x_i y_j + \sum_{0\leq i\leq n-1} \beta_i x_i + \sum_{0\leq j\leq n-1} \gamma_j y_j + \delta = 0, \tag{1.13}$$

où $\alpha_{i,j}, \beta_i, \gamma_j$ et δ sont les coefficients inconnus. Pour les retrouver, l'adversaire utilise $(n+1)^2$ couples entrée/sortie (X, Y) et forme un système linéaire en les coefficients inconnus. Pour chaque couple, l'équation (1.13) a au moins les n solutions décrites par les n relations bilinéaires indépendantes déduites de l'équation (1.12). Il est possible de retrouver l'espace vectoriel engendré par les n relations bilinéaires en temps $O(n^6)$ en utilisant une élimination gaussienne.

Une fois que n relations bilinéaires indépendantes sont connues, pour déchiffrer un message $y = (y_0, \ldots, y_{n-1})$, il suffit de résoudre un système linéaire en les x_i en temps $O(n^3)$.

Enfin, notons que l'élévation à la puissance $q^\theta - 1$ dans l'équation (1.12) ajoute des solutions parasites quand il existe des racines $q^\theta - 1$-ième de l'unité dans $\mathbb{F}_{q^n}$. Dans ce cas, il existe $\mathrm{pgcd}(q^n - 1, q^\theta - 1) = q^{\mathrm{pgcd}(n,\theta)} - 1$ telles racines et la solution n'est connue qu'à un multiple près par une telle racine. La conséquence est que le système linéaire pour retrouver les x_i n'est plus inversible mais possède un noyau de dimension $\mathrm{pgcd}(n, \theta)$. Il faut alors faire une recherche exhaustive sur cet espace pour retrouver le bon message. L'adversaire peut déterminer le message de manière unique car comme le système P est inversible, il peut tester si $P(m)$ est égal au chiffré.

Ces équations bilinéaires peuvent être aussi facilement obtenues en calculant une base de Gröbner du système composé des polynômes de la clé publique. En effet, les algorithmes calculant de telles bases, effectuent dans une première étape des multiplications des polynômes de départ $y_i = P_i(x_1, \ldots, x_n)$ par les monômes x_j et recherchent des relations linéaires entre les polynômes $x_j P_i(x_1, \ldots, x_n)$ ainsi formés. Les relations bilinéaires sont exactement des combinaisons nulles de ces polynômes.

Quand le monôme f n'est plus un monôme mais est remplacé par un polynôme, de telles relations n'existent plus. Cependant, il est possible de rechercher des relations de plus haut degré en les y_i et x_j. Dans ce cas, le comportement des algorithmes de calcul de bases de Gröbner est meilleur que pour un système aléatoire de même taille. Faugère et Joux ont montré dans [102] comment inverser ces systèmes en pratique, alors que Granboulan, Joux et Stern ont analysé le comportement asymptotique de ces algorithmes d'inversion dans [144].

Attaque différentielle contre les schémas multivariés. Dans [110], avec Louis Granboulan et Jacques Stern, nous avons introduit l'idée d'utiliser la différentielle d'une forme quadratique pour obtenir de l'information sur la clé secrète. La différentielle permet de faire baisser le degré des équations.

La différentielle $Df(X, Y)$ d'une fonction quadratique f est une forme bilinéaire symétrique définie par $f(X+Y) - f(X) - f(Y) + f(0)$. Dans le cas du système quadratique f de C^*, elle s'écrit :

$$Df(X, Y) = X^{q^\theta} Y + X Y^{q^\theta}. \tag{1.14}$$

En étudiant les formes bilinéaires symétriques associées aux formes quadratiques de la clé publique, il est possible d'obtenir des équations ou propriétés caractéristiques des systèmes dérivés de C^*. L'attaque du schéma PMI [90], de la famille de C^* est détaillée dans l'annexe A et nous la présenterons après les attaques sur SFLASH.

Contributions. Dans [97], nous avons exposé avec Vivien Dubois et Jacques Stern une autre attaque contre le schéma C^* utilisant la différentielle de ce système qui permet de retrouver une matrice M de la forme $M = S^{-1}aS$, c'est-à-dire une matrice conjuguée par S de la matrice représentant la multiplication par a dans le corps fini. À partir d'une telle matrice, nous avons également montré avec Gilles Macario-Rat et Jacques Stern dans [120] comment retrouver la matrice S.

Cette attaque exploite la forme particulière de la différentielle du monôme X^{1+q^θ}. Nous avons déduit de cette différentielle un ensemble d'équations linéaires satisfaites par les inconnues des matrices de multiplications qui ont la propriété de "commuter" avec les monômes :

$$\forall\ a \in \mathbb{F}_{q^n},\ (aX)^{1+q^\theta} = a^{1+q^\theta} \cdot X^{1+q^\theta}.$$

L'absence d'une telle relation dans le cas du schéma HFE [240], où la fonction monomiale de f est remplacée par un polynôme quadratique, a pour conséquence que cette attaque ne s'applique plus.

La multiplication par $a \in \mathbb{F}_{q^n}$ agit sur les monômes de l'équation (1.14) pour donner l'équation suivante :

$$Df(aX, Y) + Df(X, aY) = \left(a + a^{q^\theta}\right) Df(X, Y).$$

Ce système est composé de n formes bilinéaires. Il est possible de calculer la différentielle d'une fonction composée $P = T \circ f \circ S$, par les applications linéaires S et T : $DP(X, Y) = T \circ Df(S(X), S(Y))$. Nous obtenons alors la relation suivante sur la clé publique :

$$DP(S^{-1}aS(X), Y) + DP(X, S^{-1}aS(Y)) = T\left(a + a^{q^\theta}\right) T^{-1} DP(X, Y). \tag{1.15}$$

Nous avons montré dans [97] que cette équation est caractéristique des systèmes issus du chiffrement C^*. La résolution des équations bilinéaires

$$DP(MX, Y) + DP(X, MY) = NDP(X, Y) \quad \forall X, Y$$

en les inconnues M et N, deux applications linéaires sur $(\mathbb{F}_q)^n$, sont les applications

$$M = S^{-1}aS \text{ et } N = T\left(a + a^{q^\theta}\right) T^{-1}. \tag{1.16}$$

Chaque équation bilinéaire donne $n(n-1)/2$ contraintes linéaires sur les coordonnées des matrices M et N. Nous avons donc besoin de seulement trois équations bilinéaires du type $DP_i(MX, Y) + DP_i(X, MY) \in \mathrm{Vect}(\{DP_j(X, Y)\}_{1 \le j \le n})$, $i = 1$ à 3, pour retrouver M et N.

Dans [120], nous avons montré avec Gilles Macario-Rat et Jacques Stern que nous pouvons retrouver la clé secrète S. En effet, la connaissance de a dans M permet alors de linéariser les équations (1.16) et de retrouver S et T. Comme M est la matrice conjuguée de la matrice représentant la multiplication par a, ces deux matrices ont même polynôme caractéristique. De plus, le polynôme caractéristique de la matrice de la multiplication par a possède a comme racine [259]. Nous pouvons donc facilement retrouver a et la clé secrète S. Une fois que S est connue, T s'en déduit. Les détails sont donnés dans l'annexe C.

Schéma de signature SFLASH. Patarin, Goubin et Courtois ont proposé de réparer C^* en le transformant en schéma de signature dans [241] en retirant des équations de la clé publique d'origine P comme l'avait proposé Shamir dans [268]. Ils obtiennent alors un nouveau système noté $\tilde{P}$. Dans ce cas, l'attaque de Patarin [238] ne s'applique plus car

l'adversaire ne peut pas obtenir n formes bilinéaires. La matrice $\tilde{T}$ est maintenant une matrice à m lignes et n colonnes, avec $m < n$.

Le signataire, pour inverser le système $\tilde{P}$, conserve la matrice T complète. Au moment d'inverser le système, il choisit $n - m$ coordonnées aléatoirement pour les équations manquantes et inverse le système P. La signature est une donnée de taille n. Le vérifieur ne peut maintenant pas tester l'égalité des m équations de $\tilde{P}$.

Contributions. Dans [97], avec Vivien Dubois et Jacques Stern, nous avons montré que s'il existe des racines $q^\theta - 1$ de l'unité dans $\mathbb{F}_{q^n}$, il est alors possible de reconstruire une clé publique équivalente à la clé publique P à partir de $\tilde{P}$. Nous avons montré que nous pouvons efficacement calculer une matrice conjuguée par S d'une matrice représentant la multiplication par une telle racine. Si nous composons la clé publique par une telle matrice, alors nous avons également montré que nous retrouvons des formes quadratiques équivalentes à celles qui ont été retirées de la clé publique. En retrouvant suffisamment de telles équations nous obtenons une clé équivalente à celle d'un système de Matsumoto-Imai. Par conséquent, l'attaque de Patarin [238] sur le système complet P ou celle décrite dans la section précédente peuvent s'appliquer.

Supposons qu'il existe une racine ξ $(q^\theta - 1)$-ième de l'unité. L'équation (1.15) s'écrit $DP(S^{-1}\xi S(X), Y) + DP(X, S^{-1}\xi S(Y)) = 0$. Il est alors possible de retrouver les matrices M, correspondant aux conjuguées des multiplications de ces racines ξ par S en résolvant l'équation en l'inconnue M :

$$DP(M(X), Y) = DP(X, M(Y)).$$

Nous pouvons voir qu'il s'agit des applications auto-adjointes des formes bilinéaires symétriques.

Une fois que la matrice M est trouvée, nous pouvons retrouver S comme précédemment ou composer la clé publique par la matrice M. Nous obtenons alors de nouvelles équations de la forme $\tilde{T} \times \xi^{1+q^\theta} \circ f \circ S$, si $M = S^{-1}\xi S$. Ces équations sont de nouvelles combinaisons des équations $f \circ S$ et donc nous reconstruisons de nouvelles lignes d'une matrice $n \times n$, équivalente à la matrice T. Nous connaissons maintenant $2m$ équations quadratiques d'une clé publique équivalente à la clé publique P. Si $n \leq 2m$, nous obtenons une clé publique équivalente à clé publique P. Sinon, nous pouvons prendre une autre matrice M indépendante de la première. La dimension de l'espace vectoriel des solutions de M est $\text{pgcd}(n, \theta)$, ce qui nous permet de reconstruire P en entier si $m \times \text{pgcd}(n, \theta) \geq n$. La reconstruction est également décrite dans l'annexe B.

Pour les paramètres concrets de SFLASH proposés dans [241], il n'existe pas de telles racines car n est un nombre premier. Dans [96], avec Vivien Dubois, Adi Shamir et Jacques Stern, nous avons décrit une autre attaque permettant de signer n'importe quel message en retrouvant une clé publique équivalente à la clé d'origine comme dans l'attaque précédente. L'attaque permet ici encore de trouver une matrice de multiplication d'un élément a du corps fini conjuguée par S. Seulement contrairement à l'attaque précédente, l'élément a n'est pas ici une racine $(q^\theta - 1)$ de l'unité car de tels éléments n'existent pas. En utilisant l'attaque [120], il est également possible de retrouver la clé secrète avec cette matrice.

Nous sommes repartis de l'équation (1.15), mais cette fois, nous ne connaissons pas $DP(X, Y)$ en entier car la clé publique $\tilde{P}$ ne contient que m équations. Nous avons alors considéré l'équation

$$DP_i(S^{-1}aS(X), Y) + DP_i(X, S^{-1}aS(Y)) \in D\tilde{P}(X, Y), \tag{1.17}$$

pour DP_1, DP_2, DP_3. L'attaque repose sur le fait que seulement trois équations sont nécessaires pour retrouver une matrice $M = S^{-1}aS$. Une équation pour un i fixé permet

d'obtenir $n(n-1)/2$ équations linéaires car les formes bilinéaires symétriques des différentielles sont à diagonale nulle. Nous avons n^2 inconnues pour la matrice M et m inconnues pour décrire l'appartenance à $D\tilde{P}(X,Y)$ pour chaque équation. En prenant 3 équations, nous avons plus d'équations, $3n(n-1)/2$, que de variables, n^2+3m et donc M est parfaitement déterminée.

Nous allons maintenant montrer que pour $m > 2n/3$, il existe des matrices conjuguées de multiplications qui satisfont trois telles équations. Considérons l'application S qui à une matrice M associe un ensemble de formes bilinéaires :

$$S(M)(X,Y) = DP(M(X),Y) + DP(X,M(Y)).$$

Notons V le sous-espace vectoriel engendré les formes bilinéaires $DP_i(X,Y)$ pour $i = 1,\ldots,n$ de dimension n et $\tilde{V}$ le sous-espace vectoriel engendré par les m formes de la clé publique tronquée de dimension m. Nous savons que les coordonnées de $S(M)$ sont dans V et la probabilité qu'une coordonnée fixée de $S(M)$ soit dans $\tilde{V}$ est $q^{-(n-m)}$. Donc nous avons une probabilité de $q^{-3(n-m)}$ que les trois premières coordonnées de $S(M)$ soient dans $\tilde{V}$. Enfin, comme il y a q^n multiplications, nous attendons qu'il existe des multiplications aient les trois premières coordonnées de $S(M)$ dans $\tilde{V}$ si $n-3(n-m) > 1$. En effet, les multiplications par $a \in \mathbb{F}_q$ ne nous intéressent pas car la multiplication par ces éléments sont des matrices diagonales avec a sur la diagonale qui sont égales à leur conjuguée par S. Enfin, nous avons étendu cette attaque pour qu'elle fonctionne quand $m > n/2$. Les détails de cette attaque sont donnés dans l'annexe B.

Contributions. Avec Gilles Macario-Rat, nous avons montré que trois équations sont suffisantes pour retrouver les matrices S et T. Cette approche est différente de celle envisagée dans les articles [97, 96] et s'intéresse au noyau de la différentielle, c'est-à-dire à l'ensemble des k tels que $DP(k,x) = 0$ pour tout x. L'application d'une ligne de la matrice T sur un vecteur peut s'écrire comme la forme linéaire $Tr(t \cdot X)$ avec t représentant un élément de $\mathbb{F}_{q^n}$. Une forme quadratique de la clé publique s'écrit sous la forme

$$DP_i(X) = Tr(t \cdot S(X)^{1+q^\theta}).$$

Nous avons besoin de faire les remarques suivantes. Il existe plusieurs clés secrètes équivalentes dans le cas de SFLASH. Nous allons en rechercher une particulière. À cause des propriétés des clés équivalentes, nous pouvons supposer qu'il y a une clé dont la première ligne correspond à la valeur $t = 1$.

Dans le cas où n est impair, nous avons montré que le noyau des formes quadratiques de la clé publique est non nul et de dimension 1 et que ses éléments vérifient $t(S(k))^{1+q^\theta} \in \mathbb{F}_q$. Il existe donc un unique élément k_P tel que

$$t \cdot S(k_P)^{1+q^\theta} = 1.$$

Il est possible d'inverser cette formule pour obtenir une équation particulière sur k_P faisant intervenir deux termes ayant des propriétés différentes. Nous pouvons écrire l'équation sous la forme $S(k_P) = t^\delta$ où δ est l'inverse de $-(1+q^\theta)$ modulo q^n-1. Puis l'équation $k_P = S^{-1}(t^\delta)$ peut s'écrire plus simplement en remarquant que

$$\delta = (q/2-1)\sum_{i=0}^{n-1} q^i + \sum_{i=(n+1)/2}^{n-1} q^{2i\theta}.$$

Nous obtenons l'équation

$$k_P = N(t)^{(q/2-1)} S^{-1} \left(\prod_{i=(n+1)/2}^{n-1} t^{q^{2i\theta}} \right)$$

où $N(t)$ correspond à la norme de t, qui est un élément de $\mathbb{F}_q$, et le second terme qui dépend de S.

Nous allons maintenant considérer un pinceau de formes quadratiques, c'est-à-dire une combinaison linéaire à coefficients dans $\mathbb{F}_q$ de P_1 et P_2, deux formes de la clé publique, $P_{x,y}(X) = xP_1(X) + yP_2(X) = Tr((xt_1 + yt_2)S(X)^{1+q^\theta})$ avec $t_1 = 1$. Nous pouvons écrire formellement qu'il existe un élément $k_{x,y}$ vérifiant l'équation :

$$k_{x,y} = N(x + yt_2)^{(q/2-1)} \cdot S^{-1} \left(\prod_{i=(n+1)/2}^{n-1} \left(x + yt_2^{q^{2i\theta}} \right) \right). \tag{1.18}$$

Le second membre de cette équation s'écrit comme un polynôme en deux variables x et y. Posons $x = 1$ et considérons le polynôme en y seulement. Ce polynôme a un terme en $N(1 + yt_2)^{(q/2-1)}$, qui est un polynôme de $\mathbb{F}_q[y]$, et un autre terme qui est un polynôme de $\mathbb{F}_{q^n}[y]$. Nous pouvons retrouver le premier terme par interpolation. Une fois que ce polynôme est connu, une racine est $-1/t_2$, ce qui nous donne la deuxième ligne de la matrice T. Enfin, en utilisant l'équation (1.18), nous pouvons obtenir l'image de S en plusieurs vecteurs en faisant varier x et y. La taille maximale de vecteurs indépendants est $(n+1)/2$ et nous pouvons donc reconstruire la moitié de l'image de S. En utilisant un autre pinceau entre P_1 et P_3, nous pouvons reconstruire l'autre partie de S.

Contributions. Nous avons développé les attaques différentielles contre les schémas multivariés avec Louis Granboulan et Jacques Stern dans [110]. L'attaque que nous avons conçue, exploite une propriété statistique du rang de la différentielle du schéma de chiffrement *Perturbated Matsumoto-Imai*, appelé PMI. Elle permet d'inverser la fonction sans avoir besoin de la clé secrète en utilisant l'attaque de Patarin contre C^*.

Plus précisément, nous avons calculé un espace vectoriel $\mathcal{K}$ sur lequel la perturbation s'annule. En découpant l'espace complet en autant d'espaces affines que la codimension de cet espace, qui est de petite taille d'après les paramètres du système, nous avons montré que nous pouvons construire des systèmes d'équations bilinéaires pour chaque espace affine. Une fois que ces systèmes sont retrouvés, pour inverser une valeur, il suffit de l'inverser tous les systèmes et vérifier à l'aide de la clé publique celui qui correspond au bon inverse.

Le schéma de chiffrement PMI a été proposé en 2004 par Ding [90] et consiste à perturber le schéma de Matsumoto-Imai [212] par une fonction quadratique aléatoire $H \circ R$, où $R : (\mathbb{F}_q)^n \to (\mathbb{F}_q)^r$ est une application linéaire de petit rang $r \ll n$, telle qu'une recherche exhaustive sur q^r est possible, et H est un système secret composé de n équations quadratiques sur r variables. La clé publique est de la forme

$$P' = T \circ (f + H \circ R) \circ S = T \circ f \circ S + T \circ H \circ R \circ S = P + T \circ H \circ R \circ S, \tag{1.19}$$

où P est la clé publique du schéma C^*. Le chiffrement consiste simplement à évaluer les équations de la clé publique P'.

La clé secrète se compose de T, S, R et de L, ensemble des couples (λ, μ) tels que λ est dans l'image de H et μ l'ensemble des valeurs telles que $H(\mu) = \lambda$. La taille de ensemble L est q^r. Le déchiffrement est plus compliqué que le schéma C^*, mais il suffit de montrer comment inverser la fonction $f + H \circ R$ au point y. L'utilisateur légitime peut essayer

tous les points de L. Il calcule $x_\lambda = f^{-1}(y + \lambda)$ pour un λ donné, et vérifie si $R(x_\lambda) = \mu$ pour un μ associé à λ. Si c'est le cas, x_λ est un inverse possible, sinon il passe à la valeur suivante de λ dans L.

Enfin, afin d'améliorer les performances, Ding a proposé d'utiliser $f(X) = X^{1+q^\theta}$ tel que $\text{pgcd}(\theta, n) = 8 > 1$. L'attaque exploite ce mauvais choix de paramètres.

Il est naturel de chercher à éliminer la perturbation dans l'équation (1.19), pour retrouver le schéma C^*, inversible par l'attaque de Patarin [238]. Pour ce faire, il est important de trouver les points appartenant au noyau de l'application linéaire $R \circ S$ de dimension $n - r$. Une fois que la direction de ce noyau est déterminée, il suffit de découper l'espace $(\mathbb{F}_q)^n$ suivant cette direction et appliquer l'attaque de Patarin sur le bon espace. Les détails de cette attaque sont donnés dans l'annexe A.

Nous allons expliquer comment détecter les éléments du noyau de $R \circ S$. Considérons l'application différentielle du système f au point k définie par,

$$Df_k(x) = f(x + k) - f(x) - f(k) + f(0). \tag{1.20}$$

Il s'agit d'une application linéaire et dans le cas où $f(x) = x^{1+q^\theta}$,

$$Df_k(x) = kx^{q^\theta} + xk^{q^\theta} = x^{1+q^\theta} \cdot \left(\frac{k}{x} + \left(\frac{k}{x} \right)^{q^\theta} \right).$$

Le noyau de l'application linéaire $Df_k(x)$ est composé des éléments $X = k/x$ vérifiant l'équation $X + X^{q^\theta} = 0$. Ce noyau contient les racines $(q^\theta - 1)$-ième de l'unité et est de dimension $\text{pgcd}(\theta, n) = 8$. La dimension du noyau de la différentielle du système f est la même pour le système de la clé publique P car la dimension est invariante par les applications linéaires et bijectives S et T.

La différentielle de P' est la somme de la différentielle de P et de la différentielle du système $T \circ H \circ (R \circ S)$:

$$DP'_k = DP_k + T \circ DH_{R \circ S(k)} \circ R \circ S,$$

d'après les formules de dérivation des applications composées. Par conséquent quand k est dans le noyau de $R \circ S$, nous obtenons $DP'_k = DP_k$ et la dimension du noyau vaut 8. Dans le cas où k n'est pas dans le noyau de $R \circ S$, la dimension du noyau de la différentielle de $T \circ H \circ R \circ S$ peut prendre n'importe quelle valeur, qui en général fait chuter cette dimension vers 3 pour une raison technique expliquée dans l'annexe A.

Nous avons donc un algorithme pour tester l'appartenance des éléments au noyau de $R \circ S$. Enfin, nous avons montré qu'à partir de ce test statistique, il est possible de reconstruire tout le noyau en exploitant la linéarité de cet espace.

Le problème IP. Le problème d'Isomorphisme de Polynômes (IP) est le problème algorithmique suivant : étant donné deux systèmes A et B de n équations quadratiques à n variables, peut-on trouver deux matrices inversibles S et T telles que

$$B = T \circ A \circ S?$$

Ce problème est celui qu'il faut résoudre pour retrouver la clé secrète de C^* à partir de la clé publique si A est le système formé des équations du monômes $f(X) = X^{1+q^\theta}$.

L'idée des algorithmes contre le problème IP consiste à retrouver la matrice S sur quelques points. Dans les cas où le système A n'est pas constitué uniquement de formes

quadratiques homogènes, l'algorithme To-And-Fro de Patarin, Goubin et Courtois [243] est assez efficace. Si A contient un terme constant $A^{(0)}$, alors $B(0) = T(A^{(0)})$ et nous connaissons la valeur de T en ce point. La deuxième remarque est que la partie linéaire du système B, notée $B^{(1)}$, s'exprime comme $T \circ A^{(1)} \circ S$ et donc nous avons $O(n^2)$ équations linéaires entre les $2n^2$ inconnues de S et T^{-1}. Comme nous connaissons T en un point, nous en déduisons S en un point. Ensuite, si nous avons de la chance, l'algorithme peut continuer. En effet, si $S(a)$ est connu, alors $B(a) = T \circ A(S(a))$ et T est connu au point $A(S(a))$ qui n'a que peu de chance d'être colinéaire à $A(0)$. Si nous pouvons retrouver S sur n points indépendants par cette méthode, alors nous pouvons résoudre le problème IP. Cet algorithme n'a pas été analysé rigoureusement par ses concepteurs et suppose que A et B contiennent des termes linéaires et constants.

Contributions. Avec Charles Bouillaguet et Amandine Véber, nous avons analysé deux algorithmes dont la complexité en temps est respectivement $\tilde{O}(q^{2n/3})$ et $\tilde{O}(q^{n/2})$ pour résoudre le problème IP alors que les algorithmes précédents ont une complexité heuristique en $\tilde{O}(q^n)$. En effet, s'il n'y a pas de terme constant, il faut deviner la valeur de S en un point au moins, ce qui coûte q^n. Notre algorithme fonctionne également dans les cas difficiles quand les systèmes sont homogènes, c'est-à-dire A et B ne sont composés que de termes quadratiques. Le premier algorithme est rigoureusement prouvé alors que le second repose sur une hypothèse que nous avons validée expérimentalement. Nos deux algorithmes utilisent un paradoxe des anniversaires.

La différentielle du système quadratique B au point c est définie par

$$DB_c(x) = B(x + c) - B(x) - B(c) + B(0).$$

Cette application est linéaire et en utilisant les formules de dérivation de fonction composées, nous avons l'équation $T^{-1} \circ DB_x = DA_{S(x)} \circ S$. Il est facile de voir à partir de cette équation que si nous obtenons la valeur de S en un point x, alors nous en déduisons $O(n^2)$ relations linéaires supplémentaires entre les inconnues de T^{-1} et celles de S.

Si nous sommes capable de reconstruire S en n points alors nous pouvons facilement résoudre le problème IP. Dans certains cas, nous ne pouvons pas retrouver complètement S, mais seulement en deux points. Dans ce cas, il est possible de construire un algorithme hybride qui utilise les équations du système $T^{-1}B = A \circ S$ qui sont linéaires en les inconnues de T^{-1} et quadratiques en les inconnues de S, et les équations de la différentielle en ces deux points qui donnent $O(n^2)$ équations linéaires supplémentaires sur T^{-1} et S. Avec ces équations supplémentaires, nous avons montré avec Ludovic Perret et Jean-Charles Faugère dans [54] que nous pouvons retrouver S et T^{-1} en temps polynomial grâce à l'algorithme F4 [105]. Nous allons dans la suite utiliser cet algorithme comme un test d'appartenance de deux couples à l'ensemble $C = \{(x, y) \in \mathbb{F}_{q^n} \times \mathbb{F}_{q^n} : y = S(x)\}$. Cet algorithme prend en entrée A, B et deux couples de $\mathbb{F}_{q^n} \times \mathbb{F}_{q^n}$. Si les deux couples sont dans C, il retourne une solution S et T telle que $B = T \circ A \circ S$.

Nous avons donc réduit notre problème à trouver deux couples de C. La remarque principale est la suivante : si $y = S(x)$ et S et T sont inversibles, alors

$$\text{rang } DB_x = \text{rang } DA_{S(x)}.$$

Nous allons donc rechercher des couples satisfaisant cette égalité entre les rangs. Pour ce faire, prenons deux ensembles X et Y contenant des points de $\mathbb{F}_{q^n}$ choisis uniformément.

Nous allons associer à chaque élément $x \in X$, la valeur rang DB_x et à chaque élément $y \in Y$, la valeur rang DA_y. Pour trouver x et y tels que $y = S(x)$, nous allons les rechercher parmi ceux vérifiant

$$\text{rang } DB_x = \text{rang } DA_y.$$

Cependant, l'ensemble des couples (x, y) vérifiant cette égalité est trop grand. Intéressons-nous uniquement aux points de petit rang, de l'ordre de $n - \sqrt{n/3}$.

Soit $X_k = \{x \in \mathbb{F}_{q^n} : \text{rang } DB_x = n - k\}$ et $Y_k = \{y \in \mathbb{F}_{q^n} : \text{rang } DA_y = n - k\}$. Le rang de la différentielle permet de partitionner l'ensemble $\mathbb{F}_{q^n}$ en différentes classes d'équivalence, et il est clair que Y_k est l'image par S de X_k. Étudions maintenant la taille de X_k et Y_k. Vivien Dubois, Louis Granboulan et Jacques Stern ont calculé dans [98], la probabilité des rangs des différentielles de valeur $n - k$ et nous pouvons en déduire que

$$\mathrm{E}[|X_k|] = \alpha_{n,k} \cdot q^{n-k^2},$$

où $\alpha_{n,k}$ est une constante et où la probabilité porte sur le système B. Notons $\tilde{X}_k$ le sous-ensemble $X \cap X_k$ et $\tilde{Y}_k = Y \cap Y_k$. Le paradoxe des anniversaires entre $\tilde{X}_k$ et $\tilde{Y}_k$ dit qu'il suffit de prendre $|\tilde{X}_k| = |\tilde{Y}_k| = q^{\frac{n-k^2}{2}} = 2^{n/3}$ pour avoir une paire $y = S(x)$ et rang $DB_x =$ rang DA_y avec probabilité supérieure à $1/2$. Il est connu qu'une fois la "borne du paradoxe des anniversaires" dépassée, c'est-à-dire à partir de $2^{n/3}$ dans notre cas, nous pouvons trouver un second couple en prenant $2.2^{n/3}$ valeurs dans $\tilde{X}_k$ et $\tilde{Y}_k$.

Intuitivement, il y a peu de points où le rang de la différentielle est petit. Ainsi, le nombre de couples $X_k \times Y_k$ est de taille $q^{2n/3}$ si $k = \sqrt{n/3}$. En essayant tous les couples, nous allons donc trouver (x, y) tel que rang $DB_x =$ rang DA_y et $y = S(x)$ en temps $2^{2n/3}$.

Enfin, l'algorithme en temps $O(q^{n/2})$ utilise un raffinement de cet algorithme pour raffiner le test du rang en construisant des arbres de Galton-Watson. Les détails de cet algorithme ainsi que l'analyse de complexité sont donnés dans l'annexe D.

1.3 Distingueur algébrique

Dans cette section, nous décrivons des distingueurs de fonctions de chiffrement par bloc utilisant des propriétés algébriques comme les symétries ou le degré des polynômes. Le distingueur algébrique le plus célèbre concerne l'attaque sur trois tours de l'algorithme AES [84].

1.3.1 Propriétés de Symétrie

La propriété célèbre de complémentation du DES [230] se généralise à d'autres schémas en utilisant d'autres fonctions que la complémentation des entrées et permet de construire des distingueurs très efficaces.

L'objectif est de trouver des transformations sur l'espace des clés et des messages clairs et chiffrés laissant invariant le système. Soit E un système de chiffrement par bloc chiffrant un bloc de clair P sous la clé K, $E_K(P)$. La fonction de chiffrement E_K sous la clé K est *équivalente* à la fonction de chiffrement $\theta^{-1} \circ E_{\psi(K)} \circ \phi$, s'il existe des transformations inversibles non-triviales simultanément ϕ, ψ et θ telles que :

$$\forall K, P \ : \qquad \theta(E_K(P)) = E_{\psi(K)}(\phi(P)).$$

Quand la propriété précédente n'est pas satisfaite avec probabilité 1, il est souvent possible de trouver des attaques statistiques. Par exemple, pour $\phi(P) = P \oplus \Delta$ et $\theta(C) = C \oplus \Delta'$ et ψ l'identité, nous obtenons $E_K(P \oplus \Delta) = E_K(P) \oplus \Delta'$, qui est la propriété des attaques différentielles. Dans le cas où ψ n'est pas l'identité, nous obtenons alors les attaques différentielles à clés liées.

Nous pouvons également généraliser la notion de distingueur pour une fonction de compression de la même manière. Soit H une fonction de compression d'une fonction de

hachage associant à une valeur de chaînage X et un message M, une nouvelle valeur de chaînage $H(X, M)$. La fonction $H(\cdot, \cdot)$ est *équivalente* à la fonction $\theta^{-1} \circ H(\phi(\cdot), \psi(\cdot))$ s'il existe des transformations inversibles non-triviales ϕ, ψ et θ telles que :

$$\forall X, M : \qquad \theta(H(X, M)) = H(\phi(X), \psi(M)).$$

Complémentation du DES. La génération des sous-clés dans l'algorithme DES est telle que si tous les bits de la clé sont complémentés, alors toutes les sous-clés le sont également. De plus, si nous complémentons également les bits du texte clair, le chiffré est alors lui aussi complémenté. Nous avons donc la propriété suivante :

$$\forall K, P : \qquad DES_{\bar{K}}(\bar{P}) = \overline{DES_K(P)}.$$

Cette propriété permet d'accélérer la recherche exhaustive d'un facteur 2. En effet, supposons que nous ayons deux couples entrée/sortie (P, C_1) et $(\bar{P}, C_2)$ tous deux chiffrés avec la même clé K que nous cherchons à retrouver. Chiffrons P avec la moitié des clés possibles, celles dans l'ensemble $\mathtt{0}\|\{\mathtt{0},\mathtt{1}\}^{55}$ par exemple. Pour chaque chiffré C généré, nous le comparons avec C_1 et C_2. Si C collisionne avec C_1, la clé K cherchée est celle utilisée pour calculer C_1. Si C collisionne avec $\bar{C_2}$, la clé $\bar{K}$ cherchée est, celle utilisée pour calculer $\bar{C_2}$.

Contributions. Considérons le chiffrement *PURE*, schéma de Feistel pour lequel la S-Boîte est un monôme dans un corps fini. Ce schéma a été proposé par Knudsen et Nyberg dans [229] et offre une bonne résistance contre les attaques différentielles [35] et linéaires [211]. Il a été conçu avec une structure fortement algébrique. Toutes les opérations sont effectuées dans le corps fini $\mathbb{F}_{2^{32}}$. Le clair et le chiffré sont composés de deux éléments de ce corps et la clé est donnée par r éléments K_0, K_1, ... K_{r-1} où r est le nombre de tours :

$$\begin{aligned} L_0 &= P_L & R_0 &= P_R \\ L_{i+1} &= R_i & R_{i+1} &= L_i \oplus (R_i \oplus K_i)^3. \end{aligned}$$

Comme sa structure est fortement algébrique, il est naturel de rechercher des relations algébriques entre le clair et le chiffré. Dans [53], avec Charles Bouillaguet, Orr Dunkelman et Gaëtan Leurent, nous avons utilisé l'application Frobenius ($x \mapsto x^2$) dans le corps fini $\mathbb{F}_{2^{32}}$. Cette application linéaire commute avec tout monôme. Il est alors facile de vérifier que la relation de symétrie suivante est satisfaite :

$$E_K(P)^2 = E_{K^2}(P^2).$$

Plus précisément, si $K'_i = K_i^2$ pour tous les tours et si $P'_L = P_L^2$ et $P'_R = P_R^2$, alors $L'_i = L_i^2$ et $R'_i = R_i^2$ à chaque tour car $(R_i^2 \oplus K_i^2)^3 = ((R_i \oplus K_i)^3)^2$. Ceci donne un distingueur très efficace avec une attaque à clés liées.

Contributions. Dans [53], nous avons également montré le résultat suivant sur la fonction de hachage Lesamnta [152].

La fonction de compression utilise le mode de Matyas-Meyer-Oseas (MMO) [215] :

$$f(H, M) = M \oplus E_{g(H)}(M).$$

Elle est composée de deux schémas de Feistel à 4 branches en parallèle pour calculer l'expansion de clé $g(H)$, qui génère les valeurs K_i de la figure 1.1, et pour la fonction E opérant sur les valeurs X_i de la figure 1.1. Les fonctions de tour F et G de ces deux

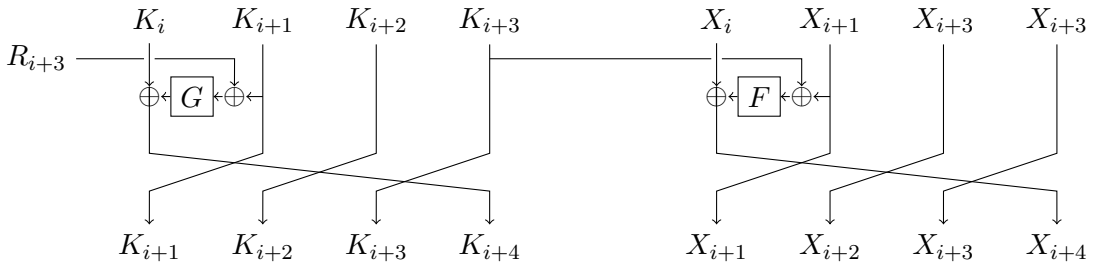

FIGURE 1.1 – La fonction de tour de Lesamnta. Cette fonction est itérée 32 fois.

opérations sont similaires à la fonction de tour de l'AES [84] et ont la même propriété d'équivalence : si les deux moitiés de l'entrée sont échangées, alors les deux moitiés de la sortie le sont également [202]. En effet, il est facile de voir que SubBytes, ShiftRows et MixColumns vérifient cette propriété.

a	b	a	b
c	d	c	d
e	f	e	f
g	h	g	h

$\xrightarrow{\text{SR}}$

a	b	a	b
d	c	d	c
e	f	e	f
h	g	h	g

et

a	b	a	b
c	d	c	d
e	f	e	f
g	h	g	h

$\xrightarrow{\text{MC}}$

a'	b'	a'	b'
c'	d'	c'	d'
e'	f'	e'	f'
g'	h'	g'	h'

Cependant, l'expansion de message de Lesamnta contient des constantes de tour R_i non symétriques pour éviter cette attaque. Malheureusement, les constantes $R_i = 2i + (2i+1) \cdot 2^{32}$ sont des mots de 64 bits symétriques excepté le dernier bit. Plus précisément, si $R_i = (R_i^\top \| R_i^\perp)$, où les parties hautes et basses de R_i sont des mots de 32 bits, alors la seule différence entre $R_i^\top$ et $R_i^\perp$ est dans le bit de poids faible.

Afin de décrire la propriété de symétrie, nous allons donner quelques notations. Deux mots x et y sont dits "échangés par moitié" si la moitié haute de x, notée $x^\top$, est égale à la moitié basse de y, $y^\perp$, et vice-versa. Nous représentons $(x^\perp \| x^\top)$ par $\overleftrightarrow{x}$. Pour prendre en compte les constantes de tour, nous définissons l'opération : $\widetilde{x} = (x^\perp \oplus 1 \| x^\top \oplus 1)$ pour laquelle nous avons $R_i = \widetilde{R_i}$. Nous étendons ces deux relations à des vecteurs de mots : $\widetilde{(x, y)} = \widetilde{(x', y')}$ si $x = \widetilde{x'}$ et $y = \widetilde{y'}$.

Dans [53], nous avons montré qu'en prenant un bloc de message m tel que $m = \widetilde{m}$, et une valeur de chaînage h telle que $h = \widetilde{h}$, alors la fonction de compression $CF(h, m)$ satisfait la propriété suivante :

$$CF(h, m) = \overleftrightarrow{CF(h, m)}.$$

Cette égalité peut être facilement vérifiée : les moitiés hautes et basses de chaque mot sont les mêmes. Par conséquent, avec un seul message, il est possible de distinguer la fonction de compression de la fonction Lesamnta d'une fonction aléatoire indépendamment du nombre de tours !

1.3.2 Propriétés des polynômes

Les attaques décrites dans cette section utilisent des distingueurs algébriques en représentant la fonction de chiffrement par une fonction polynomiale de petit degré.

Attaque par différentielle de haut degré. Les attaques différentielles de haut degré ont été proposées par Lai [201] et Knudsen [189]. Soit f une fonction définie entre deux groupes abéliens. La dérivée de f en un point a s'exprime sous la forme :

$$D_a f(x) = f(x + a) - f(x).$$

La dérivée i-ième de f aux points $a_1, \ldots, a_i$ est définie par :

$$D^{(i)}_{a_1,\ldots,a_i} f(x) = D_{a_i}\left(D^{(i-1)}_{a_1,\ldots,a_{i-1}} f(x)\right).$$

Considérons le cas particulier où f est définie de $GF(2)^m$ vers $GF(2)^n$. Dans ce cas, pour que la dérivée i-ième ne soit pas nulle, il faut que les a_i soient linéairement indépendants. La dérivée i-ième se définit à l'aide du sous-espace V de dimension i de $GF(2)^m$ sous la forme :

$$D_V f(x) = \bigoplus_{v \in V} f(x \oplus v).$$

Une propriété algébrique importante des différentielles est la suivante : si le polynôme est de degré d, alors la différentielle à l'ordre $d+1$ est identiquement nulle et la différentielle à l'ordre d est une constante. S'il existe un polynôme de degré d liant des bits du message clair à certains bits du message chiffré, alors le calcul de la différentielle à l'ordre $d+1$ de ce polynôme sera identiquement nulle. Ceci permet de distinguer le schéma de chiffrement considéré d'un schéma idéal. En effet, dans le cas d'une fonction de chiffrement idéale, le polynôme associé doit être de degré élevé, de l'ordre de 2^n, pour un chiffrement sur n bits puisque qu'il existe un unique polynôme de degré au plus 2^n passant par 2^n points.

Contributions. Dans [120], avec Gilles Macario-Rat et Jacques Stern, nous avons utilisé ces techniques pour attaquer des schémas issus de la cryptographie multivariée. Nous avons attaqué des instances d'un schéma de traçage de traîtres proposés par Billet et Gilbert dans [36]. Ce schéma est basé sur le problème de l'isomorphisme de polynômes (IP) avec des monômes de la forme $X^{1+q^{\theta_1}+\ldots+q^{\theta_{d-1}}}$ définis dans une extension $\mathbb{F}_{q^n}$ de degré n du corps fini $\mathbb{F}_q$. Ce monôme s'écrit dans une base de $(\mathbb{F}_q)^n$ en un système de n polynômes de degré d en n variables. Nous avons alors étendu les attaques différentielles sur SFLASH [97, 96] en des attaques différentielles de plus haut degré. Ceci nous a permis de retrouver la clé secrète S et T comme dans le cas de SFLASH.

Chapitre 2

Méthodes statistiques

Le but de cette partie est de décrire des attaques utilisant des invariants de nature *statistique* : *certains événements particuliers sont attendus*, comme des collisions ou des différences particulières entre les valeurs d'entrée et de sortie des fonctions de chiffrement. Ces événements se produisent avec une certaine probabilité, et l'attaquant dispose d'un test lui permettant de détecter ces événements. Dès qu'ils se produisent, l'adversaire peut obtenir beaucoup d'informations sur certains paramètres ou valeurs mis en jeu.

2.1 Collisions Internes

Les attaques que nous allons décrire dans cette section sont d'ordre générique et ne s'intéressent qu'à certaines propriétés des mécanismes. En particulier, elles n'exploitent pas les détails des fonctions. Par exemple, dans des codes d'authentification de messages ou des modes de chiffrement impliquant plusieurs primitives de chiffrement, la seule propriété de ces primitives qui sera exploitée est leur bijectivité : elles sont considérées comme des permutations aléatoires. Ces attaques reposent sur des propriétés très simples de collisions internes dans le schéma pour trouver des équations ne mettant en jeu qu'une petite partie de la clé secrète.

2.1.1 Systèmes de chiffrement symétrique

Double-DES. Le DES [230] est un système de chiffrement de 64 bits vers 64 bits résistant aux attaques différentielles et linéaires. Sa seule faiblesse réside dans la taille des clés utilisées, de 56 bits seulement. Vingt ans après son adoption, en raison de l'augmentation de la puissance de calcul des machines dédiées, la recherche exhaustive devint possible en 2 jours. Pour éviter ce problème, il a été proposé d'utiliser deux chiffrements successifs avec deux clés différentes. Cette technique est appelée *cascade*. Ce chiffrement est alors appelé 2DES :

$$2DES_{K_1,K_2}(X) = DES_{K_2}(DES_{K_1}(X)).$$

La recherche exhaustive de la clé (K_1, K_2) requiert a priori 2^{112} chiffrements. Cependant, il est possible de monter une attaque plus efficace sur ce chiffrement appelée *attaque par le milieu*. Elle consiste à séparer l'espace des clés en deux ensembles indépendants. Étant donné un couple entrée/sortie (P, C), l'idée est de trouver une *collision* au milieu du chiffrement de la forme

$$DES_{K_1}(P) = DES^{-1}_{K_2}(C). \tag{2.1}$$

Cette relation algébrique sur le chiffrement va être exploitée comme *test d'arrêt* pour une recherche exhaustive sur les clés K_1 et K_2. En effet, nous pouvons chiffrer P avec toutes les clés K possibles et ranger dans un ensemble L_1 les couples $(DES_K(P), K)$ triés selon la première coordonnée. Dans un second temps, nous calculons les déchiffrements $DES_K^{-1}(C)$ pour toutes les clés K possibles. Notons formellement cet ensemble L_2 qu'il est inutile de stocker. Une variante du paradoxe des anniversaires, nous indique alors que le nombre moyen de collision entre L_1 et L_2 sera de l'ordre de

$$\frac{|L_1| \times |L_2|}{2^{64}}$$

car une collision sur 64 bits est recherchée parmi $|L_1| \times |L_2|$ valeurs. Dans notre cas, le nombre moyen de collisions, clés (K_1, K_2) possibles, est donc $2^{56} \times 2^{56}/2^{64} = 2^{48}$ clés parmi lesquelles se trouve la bonne clé car toutes les clés vérifiant l'équation (2.1) sont générées. Les collisions sont facilement détectées car l'ensemble L_1 est trié. La recherche d'une valeur dans L_1 se fait par dichotomie en $\log_2(|L_1|) = 56$ opérations. Les autres couples qui satisfont l'équation mais ne correspondent pas à la clé recherchée sont appelées *fausses alertes*. Il faut maintenant retrouver la bonne clé dans cet ensemble L de 2^{48} clés possibles. Il faut *filtrer* l'ensemble pour retrouver la bonne clé. La seconde idée est de prendre un autre couple (P', C') et de calculer pour toutes les clés restantes celle qui appliquée sur P' donnera C'. Une mauvaise clé vérifiera cette deuxième équation algébrique avec probabilité 2^{-64}. Comme il reste $2^{48} - 1$ fausses alertes, le nombre moyen de fausses alertes vérifiant cette deuxième équation est donc $\frac{2^{48}-1}{2^{64}} < 2^{-16}$. En conséquence, avec probabilité $1 - 2^{-16}$, toutes les mauvaises clés auront été filtrées.

L'inconvénient principal de cette attaque est la mémoire nécessaire pour stocker l'ensemble L_1 : 2^{56} couples de $64 + 56$ bits. Il est possible de réduire la mémoire en utilisant un compromis temps/mémoire. En effet, en ne stockant dans L_1 que 2^m clés possibles pour $m < 56$, alors en recommençant l'attaque $2^{56}/2^m$ fois, nous générerons bien tout l'ensemble L_1. L'attaque sera donc en $2^{56+(56-m)}$ en temps et 2^m en mémoire. Pour $m = 0$, nous retrouvons la recherche exhaustive en 2^{112} sur les deux clés et pour $m = 112$, l'attaque revient à faire une recherche en table en temps presque constant.

Afin d'augmenter la complexité en temps de cette attaque, plusieurs solutions ont été proposées. La première a été proposé d'utiliser le $3DES$ avec deux clés indépendantes (K_1, K_2) en calculant

$$3DES_{K_1,K_2}(X) = DES_{K_1}(DES_{K_2}^{-1}(DES_{K_1}(X))).$$

En utilisant 3 clés indépendantes, l'attaque par le milieu a une complexité en 2^{112} en temps et 2^{56} en mémoire, et est donc hors de portée. Dans [279], van Oorschot et Wiener ont donné une attaque sur la variante 3DES avec 2 clés. Cette attaque n'est cependant pas très réaliste car elle demande d'obtenir 2^{56} couples clair/chiffré choisis parmi les 2^{64} possibles.

La deuxième proposition consiste à utiliser plusieurs modes en cascade, comme par exemple, CBC-CBC-CBC, et toutes les variantes possibles en utilisant les modes CBC, ECB, CFB, OFB. Malheureusement, tous ces modes peuvent être attaqués plus efficacement que le 3DES comme l'a étudié Biham dans [29].

Enfin, la dernière proposition pour augmenter la taille des clés et limiter l'impact de la taille petite des clés DES est la méthode du *blanchiment*. Elle est également appelée DESX et consiste à calculer

$$DESX_{K_1,K,K_2}(X) = K_2 \oplus DES_K(K_1 \oplus X).$$

Sa sécurité a été prouvée par Kilian et Rogaway [183] sous l'hypothèse du chiffrement idéal.

Attaque sur le mode CBC. Le mode CBC [215] (Cipher Block Chaining) est un mode opératoire de chiffrement randomisé et auto-synchronisant. Ce mode est sûr tant que le nombre de blocs chiffrés ne dépasse pas $2^{n/2}$ où n est la taille binaire du schéma de chiffrement par bloc. Cette borne, appelée la borne du paradoxe des anniversaires, est atteinte car il existe une attaque utilisant ce paradoxe dont la complexité est quadratique en le nombre de blocs chiffrés.

Intuitivement, la confidentialité d'un mode opératoire de chiffrement est assurée lorsqu'il est difficile de distinguer les chiffrés de deux messages choisis par l'adversaire. Cette notion de sécurité s'appelle *Left-Or-Right* (LOR) [20] pour indiquer que soit le message de gauche soit celui de droit est chiffré. Elle est équivalente à la notion Real-Or-Random (ROR) [20] qui exprime le fait que le chiffré d'un message donné est indistinguable d'une chaîne aléatoire de même taille uniformément choisie.

Dans la notion LOR, l'adversaire joue le jeu de sécurité suivant : il choisit deux messages M^0 et M^1 de même taille q blocs. Il reçoit le chiffré C de M^b et doit deviner le bit b, en retournant un bit b' avec un avantage significatif, défini par $|\Pr[b' = b] - 1/2|$. Notons que la primitive de chiffrement est considérée comme étant indistinguable d'une permutation aléatoire.

Dans le cas du mode CBC, il existe un distingueur efficace dès qu'un événement particulier, une collision, se produit. Notons n la taille de la fonction de chiffrement par bloc $E_K()$, M_i le i-ième bloc de n bits du message M^b et C_i son chiffré correspondant par $E_K()$. D'après le paradoxe des anniversaires, les collisions à la sortie de la fonction de chiffrement $C_i = C_j$ apparaissent avec probabilité strictement supérieure à $1/2$ dès que $q \approx 2^{n/2}$. Comme $E_K()$ est une permutation, si les sorties collisionnent aux instants i et j, alors les entrées de $E_K()$ aux instants i et j sont identiques et donc $M_i \oplus C_{i-1} = M_j \oplus C_{j-1}$ par définition du mode CBC. Avec forte probabilité, cette équation ne sera valable que pour un seul des deux messages M^0 ou M^1 et l'adversaire pourra alors prédire quel message a été chiffré. La probabilité de réussite de l'attaquant est de l'ordre de $q^2/2^n$.

Dans le modèle précédent, l'adversaire choisit les messages en entier avant de les soumettre. Supposons maintenant que l'adversaire choisisse le deuxième bloc des deux messages après avoir vu le chiffrement du premier bloc. Ce modèle permet de prendre en compte des situations où l'adversaire voit le début du message chiffré avant d'avoir envoyé tout le message. C'est le cas dans les protocoles de sécurité utilisé dans les réseaux de communication comme IPSEC, SSL, ou SSH, ou si le chiffrement est effectué dans une carte à puce qui n'a pas les ressources mémoires nécessaires pour stocker tout le message.

Il est alors possible d'obtenir des collisions internes plus rapidement. Sur le mode CBC, si nous connaissons le vecteur d'initialisation IV, et un couple clair/chiffré par E_K, (M_1, C_1), nous pouvons choisir le deuxième bloc du message $M_2 = C_1 \oplus M_1 \oplus IV$ pour que C_2 collisionne avec C_1 puisque les deux entrées de E_K sont les mêmes. Il est facile de voir que l'adversaire gagne le jeu de sécurité LOR ou ROR. Ce modèle de sécurité, appelé *Blockwise* par Joux, Martinet et Valette dans [167], montre certaines faiblesses des modes de chiffrement dans des situations particulières.

Contributions. Dans [122], avec Gwenaëlle Martinet et Guillaume Poupard, nous avons montré que certains modes comme le mode CFB est naturellement protégé contre ce type d'attaque. De même, une implémentation de CBC qui introduit un délai dans le chiffrement d'un bloc évite l'attaque précédemment décrite.

Dans [112], avec Antoine Joux, Gwenaëlle Martinet, et Frédéric Valette, nous avons

proposé une méthode pour sécuriser les schémas de chiffrement authentifié contre ce type d'attaque. En effet, la plupart des solutions de chiffrement et authentification ne sont pas sûres dans ce modèle car il ne faut déchiffrer le message que si l'authentification est correcte. La solution que nous avons proposée a l'avantage que le chiffrement n'est pas modifié. Le déchiffrement est *presque* au fil de l'eau, c'est-à-dire que le message est déchiffré en même temps qu'il est rechiffré de façon très simple avec un one-time-pad à l'aide d'un générateur pseudo-aléatoire. Si l'authentification est correcte, le déchiffrement retourne le germe du générateur, sinon rien, afin que si le message n'est pas correctement authentifié, l'adversaire ne puisse apprendre le message clair. Enfin, le message n'est traité qu'une seule fois par le logiciel de sécurité, ce qui est avantageux dans le cas où le déchiffrement est implanté dans un composant où la communication est pénalisante comme une carte à puce.

Enfin, dans [113] avec Antoine Joux et Guillaume Poupard, nous avons étudié les relations entre les différentes notions de sécurité dans le modèle blockwise de la même façon que Bellare, Desai, Pointcheval et Rogaway dans [21] l'ont fait pour les schémas de chiffrement à clé publique et Bellare, Desai, Jokipii et Rogaway dans [20] et Katz et Yung dans [175] pour les schémas de chiffrement symétrique. Dans le modèle Blockwise, nous avons montré que certaines notions de sécurité équivalentes avec des adversaires traditionnels, ne sont plus forcément équivalentes et nous avons donné des exemples. De même, de nouvelles difficultés apparaissent car les adversaires peuvent jouer des jeux de sécurité concurrents, ce qui n'est pas le cas dans les modèles classiques.

2.1.2 Fonction de hachage

Dans cette section, nous allons décrire diverses attaques contre les modes de construction de fonction de hachage. Certaines des attaques décrites ont sans doute contribué à la décision du NIST de lancer un appel à proposition pour construire la nouvelle fonction de hachage SHA-3. Avec Gaëtan Leurent et Charles Bouillaguet, nous avons soumis un candidat appelé SIMD [204]. Ce domaine de recherche est actuellement très actif.

Les différentes notions de sécurité pour les fonctions de hachage, résistance en collision, préimage et seconde préimage sont détaillées dans [215]. Les relations entre elles ont été étudiées par Rogaway et Shrimpton dans [258].

Attaque en multicollisions. Pour une fonction H aléatoire sur n bits, il est possible de trouver des ℓ-collisions, ℓ messages qui collisionnent, en temps $O(2^{\frac{(\ell-1)n}{\ell}})$. Pour le cas particulier du mode de Merkle-Damgård [85, 216], Antoine Joux a montré dans [163], comment construire des 2^k-collisions en temps $O(k2^{n/2})$.

Le mode de Merkle et Damgård permet de construire une fonction de hachage sur des messages de taille arbitraire en itérant une fonction de compression f transformant $n+m$ bits en n bits. Soit $M = M_1, \ldots, M_n$ le message découpé en bloc de taille $n+m$. Ce mode rajoute la longueur dans le dernier bloc ainsi que des zéros pour obtenir une taille multiple de $n+m$ et itère f à partir de la valeur initiale h_0 :

$$\forall i \geq 1,\ h_i = f(h_{i-1}, M_i) \text{ et retourne } H^f(M) = h_n.$$

La longueur dans le dernier bloc s'appelle le renforcement de Merkle-Damgård, *MD Strengthtening* et permet de montrer le théorème suivant : si la fonction de compression f est résistante aux collisions, alors la fonction de hachage H^f l'est également.

En temps $2^{n/2}$ nous pouvons trouver deux messages qui collisionnent en utilisant le paradoxe des anniversaires à partir du vecteur d'initialisation h_0. Appelons h_1 cette valeur

commune : $h_1 = f(h_0, m_1) = f(h_0, m'_1)$. Ensuite, à partir de h_1, nous pouvons trouver deux messages (m_2, m'_2) qui collisionnent : $h_2 = f(h_1, m_2) = f(h_1, m'_2)$. Les 4 messages (m_1, m_2), (m_1, m'_2), (m'_1, m_2), (m'_1, m'_2) ont tous la même image par la fonction de hachage $H^f(\cdot)$. Nous pouvons recommencer k fois cette opération pour obtenir 2^k messages distincts ayant tous la même image h_k.

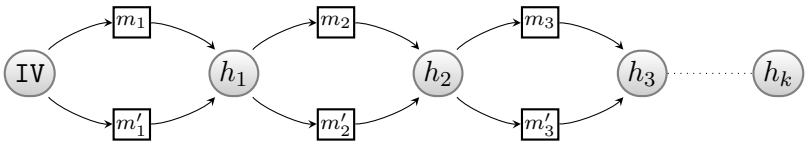

FIGURE 2.1 – Multicollisions.

Cette attaque fournit un distingueur entre une fonction de hachage aléatoire et une fonction de hachage utilisant le mode de Merkle-Damgård. Elle permet aussi d'attaquer en collision la construction :

$$H(m) = H_1(m) \| H_2(m),$$

pour laquelle une des deux fonctions, H_1 par exemple, utilise le mode de Merkle-Damgård. Supposons H_1 et H_2, deux fonctions ayant n bits de sortie. Pour la fonction H_1, nous pouvons créer une $2^{n/2}$-multi-collision en temps $O(n2^{n/2})$. Enfin, parmi ces $2^{n/2}$ messages, deux messages collisionnent pour la fonction H_2 avec probabilité supérieure à $1/2$. Ces deux messages fournissent alors une collision pour la fonction H.

Attaque en seconde Préimage. Kelsey et Schneier dans [179] ont utilisé les multicollisions pour construire une attaque en seconde préimage sur le mode de Merkle-Damgård. Une attaque similaire avait été proposé précédemment dans la thèse de Dean [87] pour certaines classes de fonctions de hachage. Cette attaque a une complexité en temps $O(2^{n-k})$ si la fonction de hachage a n bits de sortie et si le message pour lequel nous cherchons une seconde préimage est formé de 2^k blocs.

Soit M un message de 2^k blocs. Si l'encodage ne contient pas la longueur du message, c'est-à-dire sans le *MD strengthning*, il est facile de réaliser une attaque en seconde préimage. En effet, il suffit de hacher 2^{n-k} blocs N_i à partir de la valeur initiale h_0. Parmi les hachés de ces messages, l'un d'eux correspond à une valeur de chaînage dans le calcul du haché du message M, h_j par exemple. Nous pouvons vérifier que le message $N_i, M_{j+1}, \ldots, M_{2^k}$ collisionne avec le message M. En utilisant le renforcement de Merkle-Damgård, c'est-à-dire en ajoutant la longueur du message dans l'encodage, l'attaque précédente ne s'applique plus puisque qu'avec forte probabilité les deux messages n'ont pas la même taille.

Kelsey et Schneier ont étendu la construction des 2^k-collisions de Joux pour construire une famille de messages *étirables* : c'est-à-dire une famille de messages $\mathcal{M} = \{N_i\}_i$ telle que $H^f(N_i) = h_{\mathcal{M}}$ pour des messages de toutes les longueurs dans l'intervalle $[k+1, k+2^k]$. Au lieu de chercher une collision avec un message de taille un bloc pour les deux messages, il faut modifier la taille du second message. Pour la première collision, nous utilisons 1 bloc pour m'_1 et 2 bloc pour m_1 ; pour la seconde collision, 1 bloc pour m'_2 et 3 blocs pour m_2 ; pour la troisième collision, 1 bloc pour m'_3 et 5 blocs pour m_3, ... En effectuant toutes les combinaisons possibles des messages, nous obtenons la famille de messages étirables qui ont tous la même valeur de hachage $h_{\mathcal{M}}$.

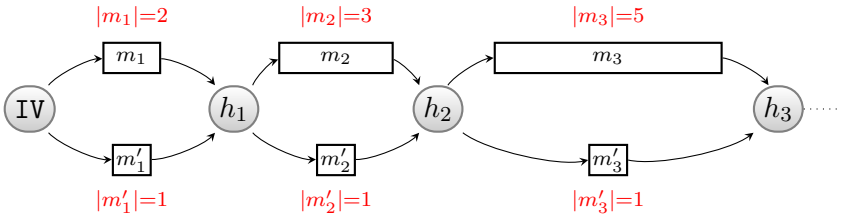

FIGURE 2.2 – Messages étirables.

La suite de l'attaque fonctionne de la manière suivante : nous choisissons au hasard 2^{n-k} fois un message $\mathbb{B}$ d'un bloc et nous calculons $f(h_{\mathcal{M}}, \mathbb{B})$ à partir de $h_{\mathcal{M}}$ la valeur de hachage commune à toute la famille de messages étirables. Si cette valeur correspond à un des 2^k hachés intermédiaires h_i pour $k < i \leq 2^k$, $f(h_{\mathcal{M}}, \mathbb{B}) = h_i$, alors le message $P, \mathbb{B}, M_{i+1}, \ldots, M_{2^k}$, où P est le message de la famille étirable de longueur $i-1$, est une seconde préimage pour $H^f(M)$. Cette attaque s'effectue en temps $O(k \cdot 2^{n/2} + 2^{n-k})$ car le coût de la génération de la famille de messages étirables est le même que celui des multicollisions, $O(k \cdot 2^{n/2})$, et le coût de la connexion est en $O(2^{n-k})$ en moyenne.

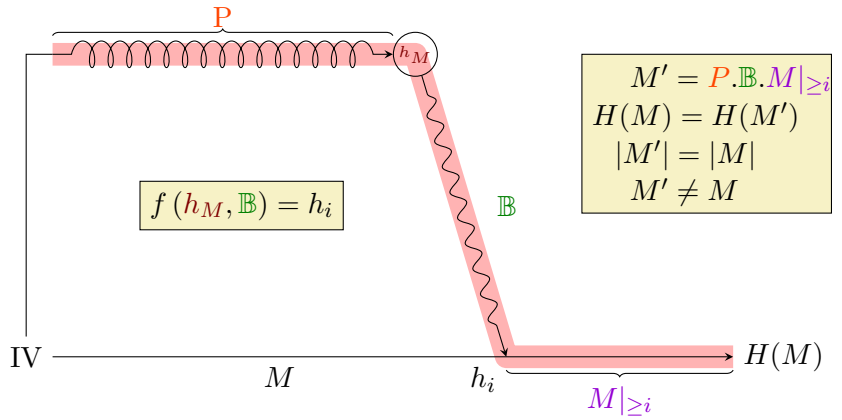

FIGURE 2.3 – Attaque de Schneier et Kelsey.

L'attaque de Dean utilise les points fixes facilement constructibles de certaines fonctions de compression et repose sur le même principe : générer une famille de messages de tailles arbitraires qui ont tous le même haché $h_{\mathcal{M}}$ et connecter $h_{\mathcal{M}}$ à l'une des valeurs de hachage du calcul de $H^f(M)$.

Nous allons décrire seulement la génération des messages. Prenons la méthode de Davies-Meyer, très répandue dans la famille MD4, pour constuire une fonction de compression à partir d'un système de chiffrement par bloc :

$$f(h, m) = E_m(h) \oplus h$$

où $E_m(\cdot)$ représente la fonction de chiffrement E sous la clé m. Pour ces fonctions de compression, il est facile de construire des points fixes : en inversant 0^n sous une clé aléatoire, message M, la valeur $h_M = D_M(0^n)$ est un point fixe de f. En effet,

$$f(h_M, M) = h_M \oplus E_M(h_M) = h_M \oplus 0^n = h_M.$$

Cependant, nous n'avons aucun contrôle sur la valeur du point fixe. En calculant $2^{n/2}$ points fixes d'une part, et d'autre part en calculant $H^f(N_i)$ pour $2^{n/2}$ messages N_i d'un bloc aléatoire, l'un d'entre eux, $f(IV, N^*)$ correspond à un point fixe $h_P = D_P(0^n)$ pour le bloc de message P avec forte probabilité. La famille de messages $N^*, P, \ldots, P$ est un message étirable car tous les messages ont le même haché h_P.

Contributions. Il existe plusieurs méthodes pour contourner ces attaques. La plus simple, appelée *double-pipe* et proposée par Lucks [207], consiste à utiliser des fonctions de compression dont la taille de l'état est le double de la taille de sortie de la fonction de hachage. C'est également ce que nous avons proposé dans la fonction SIMD [204] que nous avons soumise au concours du NIST. Il existe d'autres solutions pour contourner seulement les attaques en seconde préimage comme le mode opératoire HAIFA, proposé par Biham et Dunkelman, en rajoutant le numéro du bloc à hacher comme une entrée de la fonction de compression. Les attaques de Dean et Kelsey et Schneier ne s'appliquent plus : par exemple, dans la construction de messages étirables si nous utilisons les blocs de messages M_i^0 ou M_i^1 qui sont de tailles différentes, le compteur sera désynchronisé. En fait, nous avons montré avec Charles Bouillaguet et Sébastien Zimmer dans [66] qu'il n'existe pas d'attaques génériques sur ce mode si la fonction de compression est modélisée comme un oracle aléatoire, c'est-à-dire si ses spécificités ne sont pas prises en compte.

Autres Attaques sur le mode MD. La construction d'un ensemble de 2^k messages ayant tous le même haché en partant de 2^k valeurs initiales différentes a été proposée par Kelsey et Kohno dans [178], sous le nom de *diamant*.

Un *diamant* est un arbre binaire complet tel que la racine de l'arbre contient une valeur de haché $h_\diamond$, au niveau i, il y a 2^i hachés $h_{i,j}$ pour $1 \leq j \leq 2^i$, et il y a une arête ayant une étiquette M_u entre un noeud fils du niveau i $h_{i,j}$ et son père $h_{i-1,k}$ au niveau $i-1$ si $f(h_{i,j}, M_u) = h_{i-1,k}$.

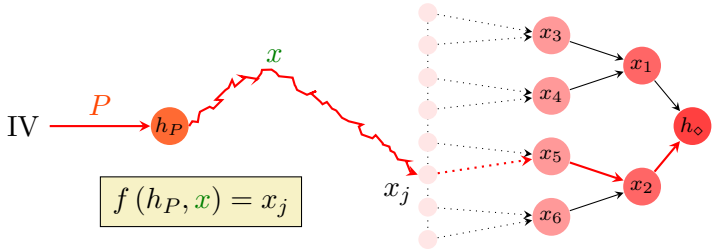

FIGURE 2.4 – Diamant.

Cet arbre binaire a 2^ℓ feuilles. En partant de n'importe laquelle des 2^ℓ feuilles, le diamant permet de trouver un message M et une valeur de chaînage $h_{\ell,i}$ tels que

$$f^{(\ell)}(h_{\ell,i}, M) = h_\diamond$$

où $f^{(i)}(h, M)$ représente la fonction de compression itérée i fois avec le message M de i blocs en partant de la valeur de chaînage h. Pour construire un tel arbre de collisions, l'idée est de partir de 2^ℓ valeurs de hachage différentes et calculer pour chaque paire, une collision et recommencer à chaque étage. La complexité est alors en $O(2^{n/2+\ell})$ car il y a $2^{\ell-1}$ paires au premier niveau, $2^{\ell-2}$ au suivant et donc $1+2+\ldots+2^{\ell-1} = 2^\ell - 1$ paires au total et le coût du calcul d'une paire est en $2^{n/2}$ en moyenne. Il est possible de faire mieux en n'imposant pas les paires qui vont collisionner au départ. Nous allons calculer le temps du passage du niveau ℓ au niveau $\ell-1$ dans l'arbre. Nous générons $2^{n/2-k}$ valeurs à partir

de chacune des 2^ℓ valeurs de chaînage présentes dans les feuilles. En tout, le nombre de paires possibles est de l'ordre de $2^{n-2k+2\ell}$ et nous demandons $2^{\ell-1}$ collisions sur n bits. Si nous voulons avoir cet événement avec forte probabilité, il faut donc que $2^{\ell-2k} = O(1)$, soit $k = \ell/2$. La complexité totale est alors en $O(\ell \cdot 2^{(n+\ell)/2})$ au lieu de $O(2^{n/2+\ell})$. Une analyse plus précise de la complexité de cette construction est donnée dans [39].

Kelsey et Kohno dans [178] ont utilisé cette structure pour monter l'attaque de Nostradamus, appelée ainsi car elle prétend prédire le futur. L'adversaire s'engage sur le résultat d'un match à venir, c'est-à-dire sur une valeur de haché h^*. Le challenger lui donne un préfixe P (le résultat réel du match) et l'adversaire doit trouver un suffixe aléatoire S lui permettant de montrer que $H^f(P\|S) = h^*$. Avec la structure de donnée en diamant, l'adversaire peut gagner l'attaque de Nostradamus en utilisant le même type d'attaque que Kelsey et Schneier [179]. En effet, étant donné P, on calcule h_P. À partir de la valeur h_P, l'adversaire essaie un bloc aléatoire x tel que $f(h_P, x)$ soit l'une des 2^ℓ feuilles de l'arbre, le coût étant $O(2^{n-\ell})$. Ensuite, à partir de n'importe quelle feuille, il peut compléter le message par un suffixe S' en suivant un chemin dans l'arbre vers la racine et donc en prenant $S = x\|S'$, $H^f(P\|S) = h^*$.

Contremesure de Rivest évitant l'attaque de Kelsey-Schneier. La contremesure de Biham et Dunkelman [232] qui consiste à rajouter le numéro du bloc à hacher n'est pas très efficace car elle réduit la bande passante d'un facteur supérieur à 10% : pour coder un compteur de 64 bits, 64 bits sur les 512 d'un bloc de SHA-1 par exemple, sont nécessaires. Rivest a proposé une technique pour réduire cette perte en utilisant le tramage, *dithering* en anglais. Le tramage est une technique de traitement d'image pour améliorer les erreurs dûes à un procédé de quantification. Pour réduire la taille du compteur, Rivest a suggéré de calculer la fonction de compression :

$$h_i = f(h_{i-1}, M_i, \mathbf{k}[i]),$$

où $\mathbf{k}[i]$ est la i lettre d'un mot infini. Il a choisi de prendre la suite de Keränen [182] qui est un mot infini sur l'alphabet $\mathcal{A} = \{a, b, c, d\}$ sans carré abélien, c'est-à-dire qu'aucun facteur de ce mot s'écrit $x\|x'$ avec x' une permutation des lettres de x. La proposition concrète de Rivest consiste à utiliser la suite infinie des lettres

$$(0, \mathbf{k}[\lfloor i/2^{13} \rfloor], i \bmod 2^{13}) \in \{0,1\} \times \mathcal{A} \times \{0,1\}^{13},$$

codées sur 16 bits, plutôt que la suite de Keränen pour générer plus rapidement les lettres.

Contributions. Dans [7], nous avons attaqué ce mode opératoire en remarquant que la complexité linéaire de la suite de Keränen considérée, c'est-à-dire le nombre de facteurs différents d'une taille donnée ℓ, est extrêmement petite. En fait, cette complexité ne peut être que linéaire dans le cas d'une suite générée par un morphisme selon un théorème dû à Cobham [72], alors que pour une suite aléatoire, la complexité est exponentielle.

Nous présentons ici une autre attaque en seconde préimage contre Merkle-Damgård sans tramage. Cette attaque est plus simple que l'attaque avec tramage, mais elle présente les principales étapes de notre attaque et en même temps, il est assez facile de l'adapter pour prendre en compte le tramage dans le cas de suites avec faible complexité en nombre de facteur. Nous avons enfin présenté une autre attaque quand cette complexité augmente en utilisant une autre structure de donnée et un compromis de Hellman [6]. Les détails sont dans l'annexe F.

Supposons que nous ayons calculé un diamant ayant 2^ℓ feuilles et permettant de créer 2^ℓ messages ayant tous la même valeur de hachés h_T en partant de 2^ℓ valeurs de chaînage différentes. Considérons un message M de 2^k blocs. Il est facile de connecter un message

d'un bloc à partir de n'importe quelle valeur de chaînage à notre diamant en $2^{n-\ell}$ et de même connecter la racine h_T à une des 2^k valeurs de chaînage en temps 2^{n-k} avec un seul bloc.

Nous pouvons connecter la racine h_T avec un message d'un bloc à une des 2^k valeurs de chaînage du calcul du message M en temps 2^{n-k}, disons h_{i_0}. Ensuite, nous savons que le diamant génère des messages de taille ℓ. Nous remontons à la valeur de chaînage $h_{i_0-\ell-1}$ et nous essayons des messages d'un bloc permettant d'atteindre une des 2^ℓ valeurs de départ du diamant. Cette étape a une complexité en $2^{n-\ell}$. Notre attaque permet en temps $2^{n-k} + 2^{n-\ell}$ de remplacer un morceau du message M de taille $\ell + 2$ par un autre message.

Pour adapter cette attaque, il suffit de prendre en compte le facteur le plus probable au moment de la conception du diamant ou de construire autant de diamants qu'il y a de facteurs. Ensuite, la complexité totale dépendra aussi de la valeur du compteur. Notre attaque a une complexité en $2^{\frac{n}{2}+\frac{\ell}{2}+2} + Fact_{\mathbf{k}}(\ell+1)\cdot 2^{n-k} + 2^{n-\ell}$ avec $Fact_{\mathbf{k}}(\ell) \leq 8\cdot\ell + 332$ représente la complexité de la suite de Keränen. Contrairement à l'attaque de Kelsey et Schneier, notre attaque demande beaucoup de mémoire pour stocker le diamant.

2.1.3 Code d'Authentification de Messages

Il existe plusieurs constructions de codes d'authentification de messages qui utilisent soit des systèmes de chiffrement par bloc, soit des fonctions de hachage, soit encore des fonctions de hachage deux-à-deux indépendantes.

Le schéma le plus célèbre est le mode CBC-MAC et consiste simplement à calculer le mode de chiffrement CBC sans IV. Si un IV est présent, il est facile de voir que l'intégrité du premier bloc de message n'est pas garantie. Soit (IV, τ) un MAC du message $M = M_1\|M_2\|\ldots\|M_k$. Un MAC valide du message $M_1'\|M_2\|\ldots\|M_k$ est (IV', τ) où

$$IV' = IV \oplus M_1 \oplus M_1'.$$

Le mode CBC-MAC n'est sûr que si l'on calcule des MACs pour des messages de taille fixe : il est facile de contrefaire un MAC valide à partir de deux MACs. Si τ est connu, le MAC du message $M = M_1\|\ldots\|M_k$, et τ' celui du message $M' = M_1'\|\ldots\|M_{k'}'$, le MAC du message $M_1\|\ldots\|M_k\|M_1' \oplus \tau\|M_2'\|\ldots M_{k'}'$ est τ'.

Il existe plusieurs méthodes pour éviter ce type d'attaques. La première consiste à rajouter la longueur du message dans le premier bloc. Ceci garantit un encodage "prefix-free" et il est possible de montrer la sécurité de ce schéma. Son inconvénient est que le MAC ne peut pas être calculé en ligne car la longueur du message doit être connue avant de commencer le calcul. La seconde méthode consiste à surchiffrer le MAC en utilisant une autre clé et s'appelle ECBC (Encrypted CBC-MAC) ou EMAC. Ce schéma a été prouvé sûr par Petrank et Rackoff dans [245].

Contributions. Dans [123], avec Gwenaëlle Martinet, Frédéric Valette et Sébastien Zimmer, nous avons étudié le schéma CCM (CBCMAC et Mode Compteur) utilisé dans la norme sans fil 802.11i. Ce schéma utilise un mode compteur pour le chiffrement et un CBCMAC surchiffré, prefix-free et randomisé.

Nous avons prouvé que quand le MAC et le schéma de chiffrement utilisent chacun une clé différente et que la fonction de chiffrement est une fonction pseudo-aléatoire, ce schéma est sûr au-delà de la borne du paradoxe des anniversaires. Ceci signifie que le schéma est sûr même si le nombre de blocs de message chiffrés et authentifiés dépasse $2^{n/2}$ où n est la taille du schéma de chiffrement par bloc. Nous avons montré que la confidentialité et l'authentification de ce schéma étaient en $L/2^n$ où L est le nombre de blocs de message.

Le MAC est un CBCMAC surchiffré où le premier bloc comprend un aléa qui ne se répète pas et la longueur du message. Nous avons également donné une attaque en $2^{n/2}$ quand les aléas se répètent, ce qui signifie que cette hypothèse est importante si nous voulons une sécurité au-delà de cette borne.

Ensuite, nous avons étudié s'il est possible de rendre ce MAC *calculable à la volée* en autorisant les aléas à se répéter et en enlevant la longueur dans le premier bloc. En effet, le surchiffrement évite les problèmes connus du CBCMAC et l'aléa initial semble une bonne idée pour augmenter la sécurité du mode EMAC au-délà de la borne du paradoxe des anniversaires. Le schéma de MAC dans ce cas est le suivant :

$$\tau = \mathrm{CBCMAC}_K(B_R \| M) \oplus E_{K'}(A_R),$$

où A_R et B_R sont deux blocs qui dépendent de l'aléa R et le MAC du message M est (R, τ). Cependant, nous avons proposé une attaque en $2^{2n/3}$ messages permettant de trouver un MAC valide quand $A_R = \mathtt{0} \| R$ et $B_R = \mathtt{1} \| R$ avec R de taille $n - 1$.

2.2 Attaque Linéaire

Les attaques linéaires sont très utilisées dans la cryptanalyse des schémas de chiffrement par bloc ou par flot. Elles reposent sur l'idée suivante : linéariser les composants non-linéaires de façon probabiliste pour obtenir des équations linéaires sur les bits de clé. Nous obtenons alors un système d'équations linéaires où chaque équation est vérifiée avec une certaine probabilité qui peut s'écrire sous la forme

$$A\boldsymbol{s} + \nu = b,$$

où A est une matrice $n \times k$, $\boldsymbol{s}$ est la clé secrète recherchée, ν le bruit ajouté aux équations qui a un biais ε et b le vecteur observé. Dans certaines attaques comme les attaques par corrélation par exemple, il est possible d'obtenir autant d'équations que l'on veut mais sans les contrôler : les lignes $\boldsymbol{a}_i$ de la matrice A sont choisies aléatoirement. Dans d'autres attaques comme les attaques linéaires contre les schémas de chiffrement par bloc, on obtient peu d'équations.

2.2.1 Learning Parity with Noise (LPN)

Le problème *Learning Parity with Noise* (LPN) est un problème fondamental en théorie de l'apprentissage [177].

Soit Ber_ε la distribution de Bernoulli de paramètre $\varepsilon \in [0, 1/2[$, telle que si $\nu \sim \mathsf{Ber}_\varepsilon$, alors $\Pr[\nu = 1] = \varepsilon$ et $\Pr[\nu = 0] = 1 - \varepsilon$, et soit $A_{\boldsymbol{s},\varepsilon}$ la distribution définie par

$$\{\boldsymbol{a} \leftarrow \{0,1\}^n; \nu \leftarrow \mathsf{Ber}_\varepsilon : (\boldsymbol{a}, \boldsymbol{a} \cdot \boldsymbol{s} \oplus \nu)\}.$$

Supposons que $A_{\boldsymbol{s},\varepsilon}$ représente également un oracle qui retourne des échantillons pris selon cette distribution indépendamment les uns des autres entre chaque tirage. Un algorithme M est dit un (t, q, δ)-algorithme pour résoudre le problème LPN_ε si

$$\Pr[\boldsymbol{s} \leftarrow \{0,1\}^n : M^{A_{\boldsymbol{s},\varepsilon}}(1^n) = \boldsymbol{s}] \geq \delta,$$

et M s'exécute en temps au plus t et pose au plus q questions à l'oracle.

Résoudre le problème LPN consiste à trouver $\boldsymbol{x}$ connaissant $\boldsymbol{a}_i$ et b_i du système suivant où les ν_i sont tirés selon une variable aléatoire ν de Bernouilli de paramètre $\Pr[\nu = 1] = \varepsilon$ et indépendant les uns des autres :

$$\begin{cases} \boldsymbol{a}_1 \cdot \boldsymbol{x} + \nu_1 &= b_1 \\ \boldsymbol{a}_2 \cdot \boldsymbol{x} + \nu_2 &= b_2 \\ \qquad\qquad ... &... \quad ... \\ \boldsymbol{a}_k \cdot \boldsymbol{x} + \nu_k &= b_k \end{cases} \tag{2.2}$$

Ce problème [1] est aussi central pour la cryptanalyse de schémas de chiffrement par flot. Nous allons voir plusieurs algorithmes pour le résoudre.

Algorithmes par maximum de vraisemblance. La première solution consiste à utiliser un algorithme par maximum de vraisemblance. Si le nombre d'équations du système 2.2 est petit, de l'ordre de $O(n)$ équations, la valeur la plus probable de $\boldsymbol{x}$ qui satisfait le plus grand nombre d'équations de ce système est la valeur $\boldsymbol{x} = \boldsymbol{s}$. L'algorithme recherche $\boldsymbol{x}$ exhaustivement et retourne celui qui satisfait le plus grand nombre d'équations. En utilisant l'inégalité de Chernoff et en sommant sur tous les $\boldsymbol{s} \in \mathbb{F}_2^n$, il est possible de montrer que l'algorithme retourne la bonne solution avec une bonne probabilité. Cet algorithme s'exécute en temps $O(n2^n)$.

La deuxième solution consiste à obtenir plusieurs équations qui ne font intervenir qu'un petit nombre de bits de $\boldsymbol{s}$. Supposons que nous ayons des équations du type $x_i = b_j$ pour plusieurs j et x_i représente le i-ième bit de $\boldsymbol{x}$, qui soient biaisées d'un facteur η. Dans ce cas, nous pouvons retrouver ce bit par un vote par majorité avec bonne probabilité dès que nous avons $C\eta^{-2}$ équations sur le même bit i. Si η est une constante, la complexité est $O(n2^n)$ pour obtenir les équations. En effet, obtenir un vecteur $\boldsymbol{a}_i$ spécifique demandent en moyenne un temps $O(2^n)$, et comme il nous en faut $O(1/\eta^2)$ pour chacun des n bits de $\boldsymbol{s}$, la complexité est en $O(n2^n)$. Contrairement à la technique précédente, le nombre initial d'équations est beaucoup plus grand pour obtenir de telles équations. Dans le cas des attaques linéaires sur les algorithmes de chiffrement par bloc, il est possible d'obtenir plusieurs fois le même vecteur $\boldsymbol{a}_i$. Malheureusement, le nombre d'équations n'est pas très grand et le biais est très petit en général.

Algorithme de Blum, Kalai et Wasserman (BKW). La troisième méthode est dûe à Blum, Kalai et Wasserman [41]. Elle ressemble à l'algorithme de Wagner [285] pour résoudre le problème k-somme que nous avons vu dans le chapitre 1. Ces deux algorithmes recherchent des combinaisons linéaires très creuses en annulant de grands paquets consécutifs de bits.

Supposons que chacune des équations est vraie avec probabilité $1 - \varepsilon$ et définissons le biais des équations $\eta = 1/2 - \varepsilon$. Si nous calculons une élimination gaussienne sur ce système, à chaque fois qu'une équation erronée est utilisée, l'erreur va alors se répercuter sur toutes les équations, même les correctes. Si l'on a une équation e_1 vraie avec un biais η_1 et e_2 avec un biais η_2, alors il est facile de voir que l'équation $e_1 \oplus e_2$ est vraie avec un biais $\eta_1\eta_2$ d'après le lemme d'empilement, Piling-Up lemma, de la cryptanalyse linéaire.

L'idée est alors de faire le moins possible de combinaisons linéaires pour avoir suffisamment d'équations vraies avec un biais suffisamment grand. L'algorithme de Blum *et al.* recherche un petit ensemble S d'équations, de taille $\sqrt{n}$ parmi les $2^{O(n/\log n)}$ équations, tel que la combinaison linéaire $\sum_S \boldsymbol{a}_i$ ait un seul bit à 1 et qu'il soit dans les $2n/\log n$ derniers

1. Il a été utilisé pour construire des schémas d'authentification symétriques efficaces sur RFID par Blum et Hopper [154] avec comme paramètre $\varepsilon = 1/4$.

bits. Pour ce faire, les bits de $\boldsymbol{x}$ sont regroupés en $\frac{1}{2}\log n$ paquets de taille $2n/\log n$. La première étape consiste à classer les équations par classe d'équivalence en fonction des $2n/\log n$ premiers bits. Ceci définit $2^{2n/\log n}$ classes d'équivalence différentes. Ensuite, une équation est choisie au hasard dans chaque classe, et elle est additionnée aux autres et éliminée. À la fin de cette étape, nous avons éliminé les $2n/\log n$ premiers bits de toutes les équations restantes et augmenté le biais de η à η^2. En recommençant ainsi de suite, nous obtenons des équations mettant en jeu uniquement les $2n/\log n$ derniers bits de $\boldsymbol{x}$.

La dernière étape effectue une recherche par maximum de vraisemblance en regardant toutes les équations ne mettant en jeu qu'un des derniers bits de $\boldsymbol{x}$. Si nous avons suffisamment de telles équations, alors un test par majorité permet de déterminer la bonne valeur de ce bit. De cette manière, les $2n/\log n$ bits de poids faible de $\boldsymbol{x}$ seront retrouvés, et en recommençant, le paquet de bits précédant le sera également, etc. Le nombre d'équations nécessaires pour que le vote par majorité fonctionne avec bonne probabilité dépend du biais des équations finales. Ce biais est une fonction exponentielle du nombre d'équations qui ont été additionnées : $\eta^{-2^{\frac{\log n}{2}}} = \eta^{-\sqrt{n}}$. Si η est une constante, la complexité en temps est $2^{-\Theta(\sqrt{n})}$, ce qui est également la complexité en nombre de requêtes et en mémoire du vote par majorité. La probabilité d'obtenir un vecteur de taille $2n/\log n$ ayant un seul 1 est $2^{2n/\log n}$ et donc le nombre de vecteurs nécessaires dans la dernière étape est $\frac{n}{\log n} \cdot 2^{-\Theta(\sqrt{n})} \cdot 2^{2n/\log n} = 2^{O(n/\log n)}$.

Afin de construire ces vecteurs, nous avons besoin de beaucoup plus de vecteurs en entrée de l'algorithme. En effet, nous avons $2^{2n/\log n}$ classes d'équivalences différentes et dans chaque classe, il faut au moins deux éléments. Comme à la dernière itération de la première phase, nous avons besoin d'autant de classes d'équivalences, nous avons besoin de $O(2^{2n/\log n})$ vecteurs.

Il n'est pas toujours facile d'obtenir autant de vecteurs surtout dans le cas des protocoles d'authentification de la famille HB. Lyubashevsky dans [208] a proposé pour engendrer les vecteurs dont a besoin l'algorithme BKW à partir de $O(n^{1+\varepsilon})$ vecteurs initiaux. Dans ce cas, pour que la probabilité de succès soit suffisant forte, la complexité de l'algorithme est en $2^{O(n/\log\log n)}$.

Contributions. Avec Éric Levieil dans [205], nous avons proposé de rendre la dernière étape un peu plus efficace. Plutôt que de collecter des équations ne mettant en jeu qu'un seul bit de $\boldsymbol{x}$ dans les $\frac{2n}{\log n}$ derniers bits, ce qui est coûteux, nous pouvons rechercher l'approximation linéaire la plus probable. Pour la trouver, la méthode naïve essaie toutes les applications linéaires et calcule celle qui satisfait le plus d'équations. Comme il y a $2^{2n/\log n}$ équations et $2^{2n/\log n}$ applications linéaires différentes, cette étape coûte $2^{4n/\log n}$. Ceci n'est pas un problème si nous ne recherchons qu'une estimation asymptotique de la complexité de l'algorithme BKW, mais ne constitue pas une avancée par rapport à BKW.

Cependant, en utilisant la transformée de Walsh-Hadamard, il est possible de trouver la meilleure approximation linéaire en temps $O(N \log N)$ avec $N = 2^{2n/\log n}$, soit en temps $\frac{2n}{\log n} \cdot 2^{2n/\log n}$. Ceci représente un avantage par rapport à la méthode de BKW en $2^{2n/\log n + \Theta(\sqrt{n})}$. Nous avons également besoin des $2^{2n/\log n}$ valeurs de la fonction $f_{\boldsymbol{s}}(\boldsymbol{a}_i) = \boldsymbol{a}_i \cdot \boldsymbol{s}$. Dans ce cas, la complexité asymptotique de l'algorithme n'est pas changée, mais un facteur constant est gagné dans la constante de la complexité sous-exponentielle de cet algorithme. Ce gain est important quand la taille des paramètres des systèmes est choisie. Les détails de l'algorithme sont donnés dans l'annexe E.

Chose, Joux et Mitton dans [70], ont également proposé d'utiliser le paradoxe des anniversaires généralisé pour les attaques rapides par corrélation. La variante avec la transformée de Fourier a aussi été proposée dans cet article.

2.2.2 Application à la cryptanalyse des systèmes de chiffrement par flot

Les attaques par corrélation sont des attaques statistiques contre les systèmes de chiffrement par flot utilisant des LFSR. Il y a plusieurs sortes de systèmes de chiffrement par flot mais deux modèles sont très populaires. Dans le premier modèle, les bits de sortie de plusieurs petits LFSRs sont combinées dans une fonction booléenne f. Dans le second modèle, un unique et grand LFSR est utilisé et certains bits à des positions fixes sont filtrés par une fonction booléenne f. À chaque itération, les LFSR avancent, et en fonction des bits de sorties ou du nouvel état, un bit d'aléa est produit. Ce bit est simplement additionné bit-à-bit au message clair pour obtenir le chiffré. La clé secrète est le contenu initial des registres.

Attaque par corrélation. Il est possible dans le premier modèle d'obtenir les équations formant le système LPN en calculant la corrélation entre la sortie des LFSR et la sortie de la fonction booléenne qui combine les différents LFSR. Dans ce cas, il est possible d'obtenir m équations d'un système LPN : le bit de sortie d'un LFSR à l'instant t s'exprime sous la forme d'un produit scalaire entre le vecteur contenant une partie de la clé secrète est un vecteur calculable si le polynôme de rebouclage est connu. L'attaque, exploitant les algorithmes par maximum de vraisemblance, a été découverte par Sigenthaler en 1985 [273] et sa complexité dépend du biais du générateur et de la longueur maximale ℓ des LFSR en $O(2^\ell/\varepsilon^2)$. Pour rendre cette attaque inefficace, il faut prendre un LFSR de très grande taille $\ell \geq 128$ ou éviter que la fonction de combinaison ait des corrélations trop grandes.

Attaque par corrélation rapide. Nous allons voir qu'il est également possible d'obtenir les équations d'un système LPN avec le second modèle de système de chiffrement par flot et cette fois, nous allons appliquer l'algorithme de Blum, Kalai et Wasserman.

Supposons qu'il s'agisse d'un registre filtré par une fonction prennant k bits du registre et retournant une valeur booléenne grâce à une fonction non-linéaire :

$$b_t = f(x_t, x_{t-\delta_1}, \ldots, x_{t-\delta_{k-1}}).$$

À tout instant, chaque valeur du registre peut s'écrire comme une application linéaire du contenu initial : $x_j = \sum_{i=0}^{n-1} c_i^{(j)} x_i$. En calculant la *meilleure approximation linéaire* L_f de la fonction de filtrage f grâce à la la transformée de Walsh-Hadamard, qui a la même complexité que la Transformée de Fourier Discrète en $k2^k$, nous pouvons obtenir les vecteurs $\boldsymbol{a}_i$, comme $L_f \circ A^{(i)}$, où A est la fonction de mise à jour du registre. Les bits de sortie peuvent alors s'écrire comme des équations linéaires des bits initiaux vraies avec la même probabilité $\frac{1}{2} + \varepsilon$ que la fonction linéaire soit égale à la fonction de filtrage. La valeur ε s'appelle le biais de l'approximation. Si nous cherchons la valeur initiale du registre, nous devons résoudre un problème LPN où le bruit ici est très élevé par rapport aux schémas du type HB de Hopper et Blum. En général, contrairement à l'algorithme de Blum *et al.* il n'est pas en général possible d'utiliser trop de paquets car le biais est petit et le coût total de cette attaque doit être inférieur à celui de la recherche exhaustive.

2.2.3 Application à la cryptanalyse des systèmes de chiffrement par bloc

La cryptanalyse linéaire permet d'approximer avec probabilité $1/2 + \varepsilon$ une fonction de chiffrement par une équation linéaire portant sur des bits de clé k, des bits du texte clair m et des bits du chiffré c :

$$k \cdot \gamma = m \cdot \delta \oplus c \cdot \delta^*,$$

en utilisant les masques δ, δ^*, γ et $\cdot$ représente le produit scalaire.

Il y a deux étapes à résoudre : la première est la collecte de ces équations et la seconde la résolution du système. Nous sommes dans le cas où il y a peu d'équations, $O(n)$, mais il est possible d'évaluer ces équations plusieurs fois. Nous avons vu que dans ce cas, le meilleur algorithme fonctionne par maximum de vraisemblance. En effet, nous pouvons demander plusieurs évaluations en changeant le message et le chiffré, ce qui permet de trouver la valeur de $k \cdot \gamma$ en $O(1/\varepsilon^2)$ évaluations, et chiffrements, si ε est le biais de l'approximation.

La première étape peut être automatisée par l'algorithme SFT comme l'a montré Vaudenay dans [283] en utilisant l'algorithme SFT de Kushilevitz et Mansour [197], analysé par Akavia dans [3], permettant de retrouver tous les coefficients de Fourier supérieurs à une fraction du plus grand coefficient. La complexité de cet algorithme est en temps $O(\mu^{-3} \log N \log\log N)$ où N est la taille de la transformée et μ le biais de la meilleure approximation pour une fonction booléenne. Cependant, la complexité de l'algorithme est une fonction exponentielle du nombre de bits de sorties de la fonction f. Il n'est donc pas possible de chercher les meilleures approximations linéaires pour toute la fonction. En pratique, ces approximations sont recherchées à la main en utilisant des techniques proches des attaques différentielles qui sont détaillées dans la section suivante. Les premières attaques ont été proposées par Matsui dans [211] contre le DES [230] et par Gilbert et Tardy-Corfdir dans [276] contre FEAL [269].

Dans une attaque linéaire, l'attaquant ne cherche pas uniquement un ensemble d'équations linéaires satisfaites par les bits de clé. Cette équation est utilisée différemment pour retrouver plusieurs bits de la clé secrète. Si une telle équation est biaisée de l'étape 1 à l'étape $r-1$ du schéma de chiffrement par bloc, l'adversaire devine les bits de la dernière clé permettant de calculer cette équation. Comme cette équation ne prend pas en compte tous les bits de sortie, seuls certains bits de clé sont nécessaires pour évaluer l'équation et une recherche exhaustive sur ces bits peut être effectuée. De plus, les mauvaises valeurs de ces bits de clé auront comme résultat que les bits de l'équation correspondant au chiffré à l'issu des $r-1$ premières étapes ne seront pas bons, mais correspondront à une valeur aléatoire. Si nous additionnons une équation ayant un biais ε et une variable aléatoire de biais nul, alors le Piling-Up Lemma dit que l'équation résultat sera de biais nul. Les bits correspondants à la bonne clé permettent eux de vérifier le biais de cette équation. Ainsi, l'adversaire est capable de calculer quelques bits de la sous-clé du dernier tour r.

L'équation linéaire biaisée nous permet de réaliser un test d'arrêt pendant une recherche exhaustive. De même dans les attaques différentielles, un test statistique, exprimé sous la forme qu'une différence fixée en entrée de la fonction de chiffrement résulte en une autre différence en sortie avec une probabilité différente de la probabilité normale, fournit également un distingueur qui permet de retrouver des bits de la dernière sous-clé par exemple de la même façon que le test linéaire le permet.

2.3 Attaque Différentielle

2.3.1 Attaque différentielle sur les fonctions de chiffrement

Les attaques différentielles sur les systèmes de chiffrement par bloc ont été découvertes par Biham et Shamir en 1990 [35]. Ces attaques exploitent des relations entre les chiffrés de deux messages clairs reliés sous la même clé. Cette technique très puissante a été étendue dans plusieurs directions : différentielle tronquée [189] par Knudsen, différentielle impossible [31] par Biham, Biryukov et Shamir, attaque boomerang [284] par Wagner. Ces attaques ont permis d'attaquer de nombreuses fonctions de chiffrement. Son application au DES [230] a montré que ce schéma est protégé contre ce type d'attaque.

2.3.2 Attaque différentielle sur les fonctions de hachage

Les premières attaques sur les fonctions de hachage remontent au milieu des années 90 par Vaudenay et Dobbertin sur MD4 [280, 93], et par den Boer et Bosselaers puis Dobbertin sur MD5 [88, 94] . En 1998, Chabaud et Joux ont proposé une attaque sur la fonction SHA-0 dans [65] qui a été améliorée par Biham et Chen en 2004 dans [33]. La première collision sur SHA-0 a été publiée dans [34].

Les travaux de Wang à partir de 2004, présentés ci-dessous, ont révolutionné la cryptanalyse des fonctions de la famille MD4 comme MD4, MD5, SHA-0 et SHA-1 [290, 292, 293, 291]. Depuis ces découvertes, la recherche dans ce domaine s'est accélérée. Dans un premier temps, la communauté cryptographique à essayer de mieux comprendre ces attaques en collision en trouvant de nouveaux chemins différentiels *à la Wang* et leurs implications sur les protocoles cryptographiques utilisant ces fonctions de hachage [203, 60, 260, 74, 288, 184, 289, 117, 168].

Ensuite, l'appel à proposition du NIST pour la nouvelle fonction de hachage SHA-3 a relancé la conception de ces fonctions en 2007. Plus de cinquante fonctions de hachage ont été proposées fin 2008, et moins d'une quinzaine ont été sélectionnées pour le second tour mi-2009. Beaucoup de nouvelles idées ont émergé comme les fonctions éponges [28] et les fonctions utilisant des briques de l'AES [191]. Avec Gaëtan Leurent et Charles Bouillaguet, nous avons proposé la fonction SIMD [204] à cette compétition. Depuis début 2009, beaucoup d'attaques ont été proposées afin d'analyser les derniers candidats.

Attaque différentielle *à la Wang*. L'absence de clé secrète dans les fonctions de hachage a pour effet que les attaques différentielles sur ces fonctions paraissent plus simples et plus puissantes que sur les schémas de chiffrement par bloc puisqu'il est possible de connaître exactement les valeurs intermédiaires.

Ces attaques fonctionnent de la même façon : dans un premier temps, un chemin différentiel ayant une bonne probabilité est recherché et dans un second temps, nous recherchons une paire de messages qui suit ce chemin. La dernière étape peut être accélérée par différentes techniques car il est possible de filtrer les messages qui s'écartent du chemin attendu sur les premiers tours.

Wang a introduit trois améliorations pour accélérer ces deux étapes :

1. le chemin est décrit avec une *différence signée* sur les bits, c'est-à-dire dans l'ensemble $\{-1, 0, 1\}$, ce qui permet de prendre en compte à la fois les différences modulaires et les différences binaires, car beaucoup de fonctions de hachage mélangent ces deux types d'opérations.
2. Une fois que le chemin différentiel est choisi, il est possible de calculer un ensemble de conditions sur l'état interne qui sont suffisantes pour que les messages M et $M + \Delta$ collisionnent.
3. Enfin, les conditions des premiers tours peuvent être satisfaites de manière déterministe avec la technique de modification de message et celles des derniers tours le sont de façon probabiliste en utilisant des messages aléatoires afin que toutes les conditions soient satisfaites.

La construction d'un chemin différentiel ayant une bonne probabilité est le cœur de ces attaques. Étant donné des différences dans la valeur de chaînage et dans le message, le chemin décrit comment les différences se propagent dans l'état à travers les étapes non-linéaires de la fonction.

Dans les attaques de Wang, les chemins différentiels sont construits en deux parties : les derniers tours de la fonction sont linéarisés, comme dans l'attaque de Chabaud et Joux, ce qui permet de trouver un chemin différentiel ayant une bonne probabilité de succès pour cette partie. Cette étape fixe les différences sur le message. Ensuite la première partie du chemin constitue la partie non-linéaire qui est plus difficile à construire pour que le chemin débute par une différence nulle dans la valeur de chaînage. La probabilité de cette partie est en général très faible.

Le chemin linéaire est construit à partir de collisions locales, introduites par Chabaud et Joux dans [65]. Ces collisions sont formées en introduisant une différence dans l'état grâce à une différence sur le message. Cette différence dans l'état est annulée quelques tours plus tard par de nouvelles différences introduites dans le message.

Enfin, même si la probabilité des premiers tours est extrêmement faible, la technique de modification de message permet de satisfaire ces conditions très simplement. Ensuite, comme la probabilité du chemin est très bonne pour les derniers tours, en essayant peu de messages, nous pouvons trouver une collision.

Contributions. Dans [117], nous avons montré avec Gaëtan Leurent et Phong Nguyen que les attaques en collision de Wang sur MD4 peuvent être utilisées pour monter des attaques sur HMAC qui retrouvent les clés secrètes.

HMAC est la principale méthode pour construire un MAC à partir d'une fonction de hachage utilisant le mode de Merkle-Damgård et dont la taille de l'état interne de la fonction est le même que celle de la sortie. Ce schéma repose sur la construction NMAC :

$$\begin{aligned} \mathrm{NMAC}_{k_1,k_2}(M) &= H^f_{k_1}(H^f_{k_2}(M)) \\ \mathrm{HMAC}_k(M) &= H^f(\bar{k} \oplus \texttt{opad} \| H^f(\bar{k} \oplus \texttt{ipad} \| M)), \end{aligned}$$

où $\bar{k}$ est la clé dérivée de k sur un bloc de taille l'entrée de la fonction de compression. HMAC est une instance de NMAC où les clés k_1 et k_2 sont déduites d'une clé k par :

$$k_1 = f(IV, \bar{k} \oplus \texttt{opad}) \quad \text{et} \quad k_2 = f(IV, \bar{k} \oplus \texttt{ipad}).$$

Nous avons étudié comment retrouver les clés k_1, k_2 pour NMAC-MD5 et HMAC-MD4. Cette attaque repose sur l'existence de chemins différentiels du type de ceux de Wang *et al.* dont la probabilité de succès dépend de certains bits de la valeur de chaînage. Nous avons besoin que cette probabilité soit assez grande, comme c'est le cas pour le chemin proposé dans [299] dont la probabilité est 2^{-58}. En effet, dans le cas des MAC, nous ne pouvons pas utiliser les techniques de modifications de message puisque la clé nous empêche de connaître les valeurs des bits. En utilisant beaucoup de messages satisfaisant le chemin différentiel, et en détectant la présence de collisions en sortie de HMAC, nous pouvons savoir si le chemin a été suivi. Ceci nous permet de déduire la valeur des bits de la valeur de chaînage, qui dans le cas de NMAC et HMAC, correspond à la clé k_2. Nous avons également proposé un algorithme qui automatise la recherche de tels chemins dans le cas de MD4.

Il est plus difficile de retrouver la clé k_1 car nous ne pouvons pas choisir tout le message. En effet, le message qui rentre dans $H^f_{k_1}()$ est la sortie de la fonction $H^f_{k_2}(M)$ sur 128 bits dans le cas de MD4 et MD5 et le reste est l'encodage du renforcement de Merkle-Damgård. Une fois k_2 connue, notre attaque a seulement besoin de construire des paires de messages dont la différence $H^f_{k_2}(M') \boxminus H^f_{k_2}(M)$ est fixée. Nous montrons comment créer ces paires en contrôlant l'encodage dans l'article décrit dans l'annexe G.

Notre attaque permet de retrouver les clés de HMAC-MD4 et NMAC-MD5 avec une complexité en temps en 2^{95} et 2^{88} messages.

Chapitre 3

Méthodes logicielles et matérielles

En plus des diverses propriétés mathématiques des fonctions cryptographiques, certaines attaques peuvent également exploiter des informations supplémentaires sur des variables internes utilisées dans les implémentations. Ces informations sont obtenues en mesurant différentes quantités physiques liées à l'implémentation elle-même, qu'elle soit logicielle ou matérielle. Les algorithmes cryptographiques sont exécutés en utilisant les composants électroniques des processeurs des ordinateurs ou des composants embarqués sur des cartes à puce ou téléphones mobiles. Les variables physiques utiles peuvent être le temps, la puissance ou le rayonnement électromagnétique émis par le composant. Il est possible de déduire de ces variables des informations sur des *valeurs intermédiaires* des algorithmes ou sur le type d'*instructions exécutées*. Dans ces attaques, l'attaquant a besoin d'avoir une description précise des algorithmes afin de connaître les valeurs intermédiaires calculées ou les instructions. Enfin, ces attaques permettent également de retrouver quels algorithmes de chiffrement sont utilisés ce qui est important dans le cadre du *reverse engineering* d'un logiciel car la consommation de puissance peut représenter une *signature* de l'algorithme.

Le terme d'attaques par canaux auxiliaires est employé pour désigner toutes ces attaques qui obtiennent des informations supplémentaires en mesurant les variables physiques. La cryptanalyse par canaux auxiliaires est un nouveau domaine de recherche de la cryptanalyse appliquée qui connaît un intérêt grandissant depuis le milieu des années 90. La recherche dans ce domaine a montré qu'une fuite d'information de nature physique, non souhaitée, durant l'exécution d'algorithmes mathématiquement sûrs pouvait avoir des conséquences dramatiques sur la sécurité comme par exemple l'extraction de la clé secrète. Ces attaques réutilisent les outils algébriques ou statistiques vus dans les chapitres précédents. Par exemple, les statistiques sont très utiles pour extraire une clé secrète à partir de mesures corrélées à une clé secrète. De même, s'il est possible d'apprendre des bits de l'aléa utilisé dans la signature DSA comme dans [223], alors les algorithmes de réduction de réseau peuvent être utilisés pour retrouver la clé secrète [225].

3.1 Attaques par mesure du temps

La première attaque physique proposée en 1996 par Kocher [195] consiste à exploiter la corrélation observée entre le temps de calcul d'une fonction cryptographique et les bits de la clé secrète manipulée. La mesure du temps de calcul peut être simplement réalisée en ayant accès au composant cryptographique en fonctionnement.

Attaque sur Square-and-Multiply. L'attaque que nous allons décrire demande d'avoir

accès au composant de déchiffrement temporairement. Elle permet de retrouver la clé privée, ce qui permet ensuite de déchiffrer tous les messages.

Considérons l'algorithme d'exponentiation modulaire allant de la droite vers la gauche et $d = d_n d_{n-1} \dots d_0$ la décomposition binaire de d : $d = \sum_{i=0}^{n} d_i 2^i$ avec $d_i \in \{0, 1\}$. L'algorithme d'exponentiation rapide, aussi appelé *square-and-multiply* calcule $M = C^d \bmod N$ avec n élévations au carré et au plus n multiplications en utilisant la relation $M = C^{\sum_i d_i 2^i} \bmod N = \prod_i C^{d_i 2^i} \bmod N$. Il est décrit dans l'algorithme 1.

Algorithme 1 Exponentiation - Square-and-Multiply : EXPBIN(C, d, N)

Require: 3 entiers positifs C, d, N.
Ensure: La valeur de $C^d \bmod N$.

1: $z = C$, $M = 1$
2: **for** $i = 0$ à n **do**
3: **if** $d_i = 1$ **then**
4: $M = M \cdot z \bmod N$
5: **end if**
6: $z = z^2 \bmod N$
7: **end for**
8: **return** M

Il est par exemple possible de retrouver la clé privée RSA presque immédiatement sur cette implémentation de l'algorithme Square-and-Multiply. Durant la première étape, l'adversaire a accès à la carte ou à une carte ayant les mêmes caractéristiques et peut lui soumettre des messages de son choix. Cette étape de *profiling* de la carte, permet à l'adversaire de mesurer le temps t_i de calcul de la multiplication $C_i \times C_i^2 \bmod N$ pour différentes valeurs de C_i. Le temps d'exécution de l'algorithme de réduction modulaire étant fonction de la valeur de C_i, les temps obtenus sont variables.

Dans la seconde étape, l'adversaire cherche à obtenir le secret bit d par bit en commençant par les bits de poids faible. Nous savons que $d_0 = 1$ car d est impair. Pour apprendre le bit d_1, l'adversaire demande le chiffrement de messages $C_1, \dots, C_k$ et mesure le temps de calcul $T_1, \dots, T_k$ nécessaire au déchiffrement complet. La seconde itération de la boucle **for** calcule $M \cdot z = C \times C^2 \bmod N$ si $d_1 = 1$ et sinon cette multiplication n'est pas effectuée. Si $d_1 = 1$, les ensembles de valeurs $\{t_i\}$ et $\{T_i\}$ seront donc corrélés. En effet, si pour un i donné, la valeur de t_i est bien plus grande que la moyenne, alors T_i est lui aussi plus grand que la moyenne. Si $d_1 = 0$, les deux ensembles sont des variables aléatoires indépendantes. En mesurant la corrélation, l'adversaire apprend le bit d_1. Puis, il recommence sur les bits suivants.

Pour se protéger contre cette attaque, il existe plusieurs méthodes de *randomisation* : elles consistent à masquer certaines données avec des valeurs variables, empêchant ainsi l'adversaire de retrouver les véritables données. Une première méthode consiste à randomiser l'*exposant* de façon à modifier les bits de d à chaque exponentiation en ajoutant à d un multiple de $\varphi(N)$. La seconde approche randomise le *message* C à déchiffrer. Avant de déchiffrer C, le composant génère un aléa R et calcule $r = R^e \bmod N$. Elle déchiffre le message randomisé $C' = C \times r \bmod N$ pour retrouver $M' = M \times R \bmod N$ puisque $C' = (M \times R)^e \bmod N$. Enfin, la randomisation est éliminée en calculant $M = M'/R \bmod N$.

Il existe peu d'exemples d'attaques par mesure du temps, à part sur les algorithmes RC5 [146] et GPS [63]. Des attaques récentes efficaces par mesure du temps utilisent le fonctionnement de la mémoire cache.

Attaque par temps utilisant la mémoire cache. Sur les processeurs multi-tâches, le temps de calcul n'est pas seulement lié au processus réalisant un calcul cryptographique. Il est possible de monter des attaques exploitant les algorithmes d'amélioration des performances des processeurs comme la mémoire cache [50, 235, 233], le cache d'instruction [1], ou les prédictions de branchement [2].

Les algorithmes cryptographiques utilisant des tables de taille importante pour calculer une fonction sont susceptibles d'être attaqués par une attaque exploitant la mémoire cache, *cache timing attack*. C'est le cas des implémentations rapides de l'algorithme AES qui utilisent une dizaine de tables où sont stockées différentes valeurs précalculées, évitant ainsi d'avoir à effectuer les opérations sur le corps fini. Ces attaques reposent sur le fait que les indices des valeurs recherchées dans ces tables dépendent de la clé secrète.

La mémoire cache fonctionne de la manière suivante : quand le processeur cherche une valeur dans une table située dans la mémoire vive, il en profite pour stocker cette valeur *ainsi que les valeurs suivantes* dans une mémoire plus rapide d'accès que la mémoire vive, de petite taille, appelée la mémoire cache. Cette méthode repose sur le principe suivant : si nous avons eu besoin de lire une valeur dans une table, nous aurons sûrement besoin d'accéder aux valeurs suivantes dans peu de temps. Par conséquent, si le programme accède à une donnée puis à une seconde donnée et que l'attaquant s'aperçoit que la seconde lecture a été rapide, il en déduit que les accès sont situés à des adresses voisines. Par conséquent, les deux adresses ont les mêmes bits de poids forts. Si les adresses dépendent de la clé, cette information peut être cruciale pour l'adversaire.

Il existe de nombreux travaux sur ces attaques comme par exemple [50, 235] contre des implémentations rapides de DES et AES. En 2006, Osvik, Shamir et Tromer ont présenté une nouvelle attaque dans [233] utilisant le principe d'éviction du cache. Le cache est une zone mémoire partagée par tous les processus mais bien sûr, chaque processus ne peut lire que l'information qu'il a écrit. Supposons qu'un deuxième processus soit lancé en parallèle d'un processus de chiffrement, et que l'adversaire soit capable de contrôler le processus de chiffrement. Cette hypothèse est réaliste dans le cas des logiciels de chiffrement de disque dur car il suffit de sauvegarder un fichier sur le disque pour lancer ce processus. Enfin, l'adversaire connaît l'emplacement des tables accélérant le chiffrement dans la mémoire vive. Cette hypothèse n'est pas très forte car cette information est accessible à partir de l'exécutable.

Initialement, le deuxième processus, appelé processus d'éviction, vide le cache en lisant une table de taille suffisamment grande pour remplir toute la mémoire du cache. Ensuite, le processus de chiffrement accède à une valeur dans la table et la toute la ligne où est stockée cette valeur se retrouve dans la mémoire cache. Quand le processus d'éviction tente de relire la table, au moment où il accède la ligne qui a été remplacée par le processus de chiffrement, le temps d'accès est plus long. En effet, l'adresse d'une variable dans le cache dépend de son adresse dans la mémoire vive. L'attaquant est donc capable d'apprendre l'adresse de la table qui a été lue par le processus de chiffrement, et donc les bits de poids de l'adresse lue.

Contributions. Dans [66] avec Thomas Chardin et Delphine Masgana, nous avons conçu une attaque contre une implémentation de RC4 intégrée à la librairie Openssl utilisant ce type d'information. La table de RC4 utilise des entiers sur 32 bits pour stocker les octets pour améliorer l'efficacité de l'implémentation. Il est possible d'utiliser le principe des attaques par éviction de cache pour apprendre les 4 bits de poids forts des indices manipulées et un algorithme de propagation de croyance [209] retrouve la table.

L'algorithme de mise à la clé de RC4 remplit un tableau S contenant 256 octets représentant une permutation de $\{0, \ldots, 255\}$ et initialise deux octets i et j à 0. La génération

d'un octet chiffrant de RC4 utilise l'algorithme 2.

Algorithme 2 Algorithme de Génération Pseudo-Aléatoire RC4

1: $i \leftarrow i + 1$
2: $j \leftarrow j + S[i]$
3: échange$(S[i], S[j])$
4: $k \leftarrow S[i] + S[j]$
5: **return** $S[k]$

Notre attaque retrouve la table S et la valeur de j à un instant t donné. Ensuite, il existe des algorithmes [32] pour retrouver la clé à partir d'un état car la fonction de génération est inversible si j est connu. Pour chaque élément u de la table, nous conservons les valeurs possibles de cet octet en utilisant une table de 256×256 booléens S_{val} avec $S_{val}[u][v] =$ `faux` si et seulement si nous savons que $S_{val}[u] \neq v$. Une autre table j_{val} de 256 booléens est utilisée pour les valeurs de j avec la même sémantique. La valeur de i n'a pas besoin d'être recherchée car elle avance indépendamment de la clé secrète.

Supposons que nous connaissions les trois indices a, b et c. Nous pouvons connaître la valeur du premier indice a par exemple car l'indice i est incrémenté modulo 256 entre chaque octet de chiffrement. De plus, la valeur *out* est observée en sortie. Cependant, nous ne connaissons pas l'ordre entre (a, b, c) et (a, c, b).

L'algorithme de RC4 effectue trois accès dans la table S et impose les contraintes suivantes :

$$\begin{aligned} j + S[a] &= b \\ S[a] + S[b] &= c \\ S[c] &= out. \end{aligned}$$

À partir de ces équations, les valeurs de $S[a]$, $S[b]$ et $S[c]$ sont complètement déterminées par l'ordre de lecture, les valeurs de j et de *out*. Pour chaque valeur de j, nous vérifions si les valeurs correspondantes de $S[a]$, $S[b]$ et $S[c]$ sont encore possibles. Si elles ne le sont pas, nous sommes certains que l'ordre et les valeurs ne sont pas possibles, ce qui nous permet de réduire le nombre de candidats, sinon nous mettons à jour les valeurs de S_{val} et j_{val}.

Dans le cas du cache, nous ne connaissons pas tous les bits des indices a, b et c. Cependant, une version probabiliste de cet algorithme retrouve la table en entier avec 500 octets de sortie.

3.2 Attaques par consommation de courant

Les circuits intégrés sont construits à partir de transistors. Le mouvement des charges à l'intérieur des transistors consomme de l'énergie et produit un champ électromagnétique. La mesure de l'activité électronique globale d'un composant est une combinaison complexe de ces petits effets.

Il est possible d'obtenir de l'information sur les données manipulées en observant les courbes de consommation de courant. Les données informatiques sont stockées dans des registres qui sont physiquement réalisés avec des transistors. Quand un registre change de valeur, certains transistors passent de 1 à 0 ou de 0 à 1 et un courant proportionnel au poids de Hamming entre les deux états est émis. Par conséquent, la multiplication d'une valeur par la valeur zéro nécessite moins de puissance que la multiplication par

n'importe quelle autre valeur. De même, si nous calculons deux fois la même fonction sur les mêmes entrées, la puissance émise pendant les deux calculs est identique au bruit près, et nous observons des *collisions* dans les courbes de consommation. De plus, si une donnée secrète est utilisée plusieurs fois pour calculer une fonction, les courbes de consommation sont *corrélées* à la donnée secrète. Enfin, la puissance émise dépend de la valeur, mais aussi de l'instruction : par exemple, une addition nécessite moins de puissance qu'une multiplication. Les courbes de consommation de ces instructions sont donc plus ou moins longues dans le temps. Réciproquement, les différences, l'abscence de consommation ou les similarités entre deux courbes de consommation de puissance donnent de l'information sur les valeurs manipulées ou leur différence et sur l'instruction.

Il existe deux types d'attaques par consommation de courant : les attaques élémentaires qui ont besoin de peu de courbes de consommation et les attaques différentielles qui ont besoin de beaucoup plus de courbes.

Les attaques élémentaires reposent sur l'hypothèse que la courbe de consommation de puissance des instructions est différente d'une instruction à l'autre ou que la forme de la courbe d'une instruction donnée dépend de la valeur ou du poids de Hamming de la valeur manipulée par cette instruction.

La consommation de courant d'une instruction peut avoir une forme caractéristique. Les attaques élémentaire par mesure de consommation de puissance, *Simple Power Analysis* (SPA), permettent de savoir quelles instructions ont été exécutées [196]. Si les instructions dépendent des bits de clé comme dans certaines implémentations de l'algorithme square-and-multiply, alors il est possible de retrouver la clé.

Les autres attaques élémentaires sont capables d'obtenir de l'information sur les *valeurs manipulées*. Les attaques *templates* [68] permettent d'avoir de l'information sur la valeur manipulée à partir de l'amplitude de la courbe. Les attaques Valeur Zéro sont un cas particulier des attaques Template quand la valeur zéro a une amplitude caractéristique [142]. Les attaques par collision [266] permettent d'apprendre qu'une fonction a été exécutée deux fois avec la même valeur en entrée.

Enfin, les attaques différentielles, *Differential Power Analysis* (DPA) [196, 67, 143, 76, 217], reposent sur l'hypothèse fondamentale que des variables intermédiaires ne dépendent que de peu de bits de la clé secrète. Dans ce cas, des méthodes statistiques comme la différence des moyennes [196] ou des tests de corrélation [57] permettent de déterminer l'hypothèse correcte entre les valeurs observées et les valeurs calculées à l'aide d'un modèle de consommation et de la valeur qui dépend de la clé [210].

3.2.1 Attaques élémentaires par mesure de courant

Les attaques SPA permettent de déduire des informations sur les instructions exécutées à partir d'un petit nombre de courbes de consommation de courant.

Dans les algorithmes où certaines instructions sont effectuées en fonction de bits de clé, comme l'algorithme Square-and-multiply, il est possible d'appliquer ce genre d'attaque. Si l'implémentation n'est pas protégée, il est possible de lire directement la clé secrète RSA sur une seule courbe de consommation de courant.

Il est facile de se protéger contre ce type d'attaque en effectuant toujours les mêmes opérations indépendamment des valeurs de bits de clé. Par exemple, l'implémentation de Montgomery est naturellement protégée. Une autre implémentation consiste simplement à toujours faire la multiplication, et à ne prendre en compte le résultat que si elle était nécessaire, et s'appelle *square-and-multiply-always*.

Algorithme 3 Square-and-Multiply-Always - EXPBINALWAYS(C, d, N)

Require: 3 entiers $C, d, N \geq 0$.
Ensure: La valeur de $C^d \bmod N$.

1: $z = C$, $M = 1$, $N[2]$
2: **for** $i = 0$ à n **do**
3: $\quad N[0] = M$
4: $\quad N[1] = M \cdot z \bmod N$
5: $\quad M = N[d_i]$
6: $\quad z = z^2 \bmod N$
7: **end for**
8: **return** M

Contributions. Avec Sébastien Kunz-Jacques, Gwenaëlle Martinet, Frédéric Muller, et Frédéric Valette, nous avons proposé une attaque SPA contre l'algorithme square-and-multiply par fenêtre glissante dans [114].

Dans le cas de l'implémentation par fenêtre glissante, une attaque SPA ne permet pas toujours de déduire tous les bits de la clé secrète car quand la fenêtre F_i est non nulle, nous ne savons pas exactement la valeur de cette fenêtre. En revanche, nous savons d'après l'algorithme de découpage, que cette fenêtre commence par un 1 dans le cas des fenêtres glissantes et qu'elle se termine par un 1 dans le cas des fenêtres de taille variable. En moyenne, 40% des bits peuvent être appris, mais ces bits ne sont pas consécutifs et les algorithmes à base de réduction de réseau ne sont pas efficaces dans ce cas. Nous avons alors conçu une attaque utilisant le principe *guess-and-determine* que nous avons décrite dans le chapitre 1. Les détails de cette attaque sont donnés dans l'annexe J.

Algorithme 4 SLIDINGWINDOW(C, d, N, m)

Require: 4 entiers $C, d, N, m \geq 0$.
Ensure: La valeur de $C^d \bmod N$.

1: [Précalcul] Calculer et stocker $C^w \bmod N$ pour tout $w \in \{1, \ldots, 2^m - 1\}$
2: Couper d en fenêtre nulle et non-nulle F_i de longueur $L(F_i)$ d'au plus m bits pour $i = 0, 1, \ldots, k-1$. Ce découpage peut être par fenêtre de taille constante ou variable.
3: $M = C^{F_{k-1}} \bmod N$ (qui est une valeur précalculée.)
4: **for** $i = k - 2$ à 0 **do**
5: $\quad M = M^{2^{L(F_i)}} \bmod N$
6: $\quad$ **if** $F_i \neq 0$ **then**
7: $\quad\quad M = M \times C^{F_i} \bmod N$
8: $\quad$ **end if**
9: **end for**
10: **return** M

Contributions. Dans [121], nous avons étudié l'algorithme de Garner pour calculer $s \bmod N$, étant donné $s_p = s \bmod p$ et $s_q = s \bmod q$. Contrairement à l'algorithme classique de Gauss décrit dans [272], l'algorithme de Garner n'a besoin que de calculer modulo les facteurs p ou q et jamais modulo $N = pq$. Supposons que $u = q^{-1} \bmod p$ soit précalculé, et posons $t = s_p - s_q$. Si $t < 0$, l'algorithme effectue en plus l'instruction $t \leftarrow t + p$. Enfin, dans tous les cas, $s = s_q + (t \cdot u \bmod p) \cdot q$.

Avec une attaque SPA, il est possible de détecter si l'addition $t \leftarrow t + p$ est exécutée.

En l'absence d'addition, nous en déduisons que $s_q < s_p$. Si nous supposons en outre que $p < q$, alors nous avons $s_q < s_p < p < q$. Enfin, si les facteurs p et q sont déséquilibrés, $|q| - |p| > \ell$, chaque fois que nous détectons qu'une addition n'a pas été effectuée, nous obtenons une signature s telle que

$$s = s_q + (t \cdot u) \bmod q \text{ avec } |s_q| < q/2^\ell.$$

Le résultat donné au chapitre 1 permet alors de factoriser N dès que nous avons environ 40 signatures qui vérifient ces contraintes. Il faut environ $40 \cdot 2^\ell$ générations de signature pour les obtenir. La taille de ℓ dépend de la taille du module N, et nous avons montré que l'attaque fonctionne avec $\ell = 11$ pour des modules de 1024 bits. Les détails de ce résultat sont décrits dans l'annexe I.

3.2.2 Attaque élémentaire par rayonnement électromagnétique

Contributions. Dans [126], avec Denis Réal, Frédéric Valette et M'hamed Drissi, nous avons développé une attaque consistant à observer la *retenue dans une addition*. Les calculs en cryptographie à clé publique travaillent généralement sur des données de très grandes tailles qui ne peuvent pas être stockées dans un mot machine. Lors d'une addition par exemple, la retenue doit se propager d'un mot à l'autre. Si la retenue n'est pas protégée, il est possible de l'obtenir par SPA si nous savons exactement dans quel registre elle est stockée.

Nous utilisons ici une attaque par rayonnement électromagnétique pour prendre en compte l'effet physique *local* à ce registre, alors qu'en général les attaques par consommation de puissance n'observent que la puissance consommée globalement par tout le circuit.

Nous avons utilisé cette information pour retrouver la clé privée de signature ECDSA [227] S lors de l'opération de randomisation de la clé secrète quand un multiple du nombre de points de la courbe $\#\mathcal{E}$ est ajouté. Dans ce cas, la clé secrète S est ajoutée plusieurs fois à des valeurs différentes et inconnues. Supposons que cette clé soit découpée en mots de ℓ bits. Nous la retrouvons mot par mot en commençant par le mot de poids faible. Au début, nous recherchons S_0 et nous observons la retenue de l'opération $S_0 + R$ où R est une variables aléatoire inconnue.

Nous avons montré que la retenue de cette addition dépend de la valeur de S_0. Par exemple, si S_0 vaut `0Xffff...ffff`, la probabilité d'obtenir une retenue sera plus grande que si S_0 vaut `0X0000...0000`.

Nous pouvons en général apprendre les bits de poids forts de chaque mot de la clé. Cette attaque nécessite d'avoir beaucoup d'expériences pour réussir. Nous avons calculé la probabilité conditionnelle d'un mot S_i de S connaissant le nombre de retenues. En prenant un intervalle de taille $2^\ell/\sqrt{m}$ autour de la valeur du pic si m est le nombre d'expériences, avec probabilité 0.99, la valeur de S_i se trouve dans cet intervalle. Enfin, cette attaque nécessite d'être combinée à une recherche exhaustive pour retrouver les bits manquants de S.

Nous pouvons remarquer que cette attaque considère une opération effectuée pour éviter les attaques différentielles ou par mesure du temps. La mise en œuvre des contremesures est délicate car leurs implémentations doivent aussi être protégées.

3.2.3 Attaque Template

Les attaques par formes, *templates* [68] sont une variante des attaques SPA pour lesquelles la forme des courbes de consommation de courant donnent d'autres informations que l'instruction effectuée. Ces attaques prennent aussi en compte les valeurs des opérandes. Il a été observé que la forme ou plus précisément l'amplitude des courbes de consommation de courant ou le rayonnement électromagnétique d'une instruction dépend de l'instruction mais aussi du *poids de Hamming de l'opérande*.

Contributions. Dans [118], avec Gaëtan Leurent, Denis Réal et Frédéric Valette, nous avons monté une attaque template sur une implémentation embarquée de HMAC-SHA-1. Nous nous sommes aperçu que le rayonnement électromagnétique de certaines instructions dépend du poids de Hamming des valeurs manipulées. Dans cette attaque, nous cherchons la valeur de K de 128 bits découpée en 4 mots de 32 bits. Nous avons étudié le calcul de la fonction de compression sur $K \oplus \texttt{ipad}$ et sur $K \oplus \texttt{opad}$ pendant le calcul des deux sous-clés de HMAC-SHA-1. L'implémentation de `xyssl` utilise 12 chargements `lwd` pour chacun des mots de 32 bits avec des valeurs différentes mais toutes reliées à un mot de K. Pendant chaque chargement, nous pouvons apprendre le poids de Hamming de la valeur chargée. En fait, il y a 6 chargements pour traiter chacune des deux sous-clés. Comme en moyenne le poids de Hamming d'un mot de 32 bits donne 2,79 bits d'information, nous avons juste assez d'information, $12 \times 2.79 = 33.48$ bits en moyenne pour retrouver chaque mot de 32 bits de K. Notre algorithme effectue une recherche exhaustive sur la valeur de ces mots qui satisfont le mieux les poids de Hamming observés en utilisant un algorithme par maximum de vraisemblance. Les détails sont donnés dans l'article [118].

3.2.4 Collision

Ce type d'attaque repose sur l'hypothèse qu'il est plus facile de détecter dans des courbes de consommation de puissance des formes identiques sur de longues périodes de temps que sur de courbes périodes. En effet, dans une SPA, nous recherchons les formes des instructions. La forme dépend de l'instruction mais aussi du poids de Hamming de l'opérande. Si l'opérande est le même, alors la forme de l'instruction est plus facile à détecter. Dans de nombreux algorithmes, dès qu'une collision sur une valeur apparaît, pendant de nombreuses instructions, les courbes vont être similaires. C'est le cas par exemple dans l'algorithme DES quand la partie droite collisionne, les courbes de consommation du calcul de la fonction de tour sont identiques, au bruit près.

Contributions. Avec Frédéric Valette dans [128], nous avons utilisé ce type d'attaque contre deux randomisations de l'algorithme Square-and-Multiply proposée par Jean-Sébastien Coron contre les attaques DPA dans [76] quand l'algorithme d'exponentiation binaire calcule de gauche à droite, c'est-à-dire des bits de poids fort vers les bits de poids faible. Cette implémentation, décrite dans l'algorithme 5, est parfois préférée car elle n'a pas besoin d'un registre supplémentaire.

Nous avons observé que si nous calculons le chiffrement de C et celui de $C^2 \bmod N$ avec l'algorithme square-and-multiply de gauche à droite, il est possible d'obtenir beaucoup de collision sur les valeurs intermédiaires du calcul. Par exemple, nous observons que la valeur obtenue après une élévation au carré à l'étape i dans le calcul de $C^d \bmod N$ est la même que celle obtenue dans l'élévation au carré calculée à l'étape $i-1$ dans le calcul de $C^{2d} \bmod N$ si et seulement si $d_{i-1} = 0$. Notons $L_j = \sum_{i=j}^{n} d_i 2^{i-j}$. Alors $P_j(C) = C^{L_j} \bmod N$ qui est

le contenu du registre M dans l'algorithme 5 après $n-j$ itérations. En effet, nous avons :

$$\begin{aligned} P_j(C) &= \prod_{i=j}^{n} C^{d_i 2^{i-j}} \bmod N \\ &= C^{d_j} \times \prod_{i=j+1}^{n} C^{2(d_i 2^{i-j-1})} \bmod N \\ &= C^{d_j} \times P_{j+1}(C^2) \end{aligned}$$

Par conséquent, si $d_j = 0$, alors $P_j(C) = P_{j+1}(C^2)$ et donc, nous pouvons retrouver les bits 0 de la clé secrète d, ce qui donne tous les bits de d.

Algorithme 5 ExpbinLR(C, d, N)

Require: 3 entiers $C, d, N \geq 0$.
Ensure: La valeur de $C^d \bmod N$.

1: $M = 1$
2: **for** $i = n$ à 0 **do**
3: **if** $d_i = 1$ **then**
4: $M = M \cdot C \bmod N$
5: **end if**
6: $M = M^2 \bmod N$
7: **end for**
8: **return** M

Cette remarque permet d'attaquer en SPA la contremesure DPA consistant à randomiser l'exposant de déchiffrement avec un multiple de $\varphi(N)$ ainsi que la randomisation des messages car avec une seule courbe, nous pouvons retrouver l'exposant secret. Les détails de cette attaque sont donnés dans l'annexe H.

3.3 Attaques par Fautes

Un autre type d'attaque physique consiste à générer des fautes pendant le calcul. En effet, il est possible d'utiliser des phénomènes physiques pour perturber le comportement du processeur pendant le calcul d'une fonction. Les instructions sont exécutées au rythme d'une horloge. En accélérant l'horloge, le temps d'exécution d'une instruction peut se trouver réduit à un point où les valeurs en sortie des bascules ne sont pas stables car les condensateurs ne se sont pas correctement chargés, et le résultat de l'instruction est imprédictible. De même, nous pouvons créer une perturbation électrique pendant le calcul d'une instruction en utilisant un flash de lumière. L'énergie dégagée par le rayon lumineux est absorbée par les électrons du silicium qui vont se déplacer. Un courant électrique, appelé photoélectrique local est alors créé et les portes logiques et les cellules mémoires réagissent à ce courant. Une autre méthode consiste à diminuer la tension d'alimentation d'un composant en dessous de la plage de fonctionnement afin de perturber le comportement du composant. Dans ces conditions anormales, le comportement physique est altéré et des erreurs de calcul sont possibles. Si l'adversaire est capable d'avoir physiquement accès au processeur, il peut alors injecter des fautes pendant le calcul. Il est difficile de savoir quel type de fautes en résulte et une étape de modélisation est nécessaire. Par exemple, les fautes sont-elles aléatoires ou ont-elles comme conséquence que le registre ne contient que des bits à 1 ? S'étendent-elles sur plusieurs bits ou sont-elles localisées sur le bit de poids

fort ? Est-il possible de réaliser plusieurs fois la même erreur ? La première étape dans une attaque par faute consiste donc à caractériser le modèle de faute, c'est-à-dire ce que fait la faute.

Attaque sur RSA-CRT. Ce type d'attaque a été découvert par Boneh, DeMillo et Lipton dans [45] et l'exemple le plus célèbre en cryptographie à clé publique concerne l'implémentation de RSA qui utilise le théorème des restes chinois [215]. Pour signer un message, on commence par calculer la signature s modulo les deux facteurs p et q. Ensuite, le théorème des restes permet de combiner $s \bmod p$ et $s \bmod q$ pour retrouver $s \bmod N$ en utilisant l'algorithme de Garner par exemple [215]. Cette implémentation permet de gagner presque un facteur 4 sur le temps de calcul.

Supposons qu'une erreur soit injectée pendant le calcul de $s \bmod q$ par exemple, aucune pendant celui de $s \bmod p$, et que nous ayons une signature du même message calculée sans erreur. Nous supposons également que la fonction d'encodage est déterministe. La signature erronée $\tilde{s}$ et la signature correcte s vérifient les équations $s = \tilde{s} \bmod p$ et $s \neq \tilde{s} \bmod q$. Le calcul suivant nous permet d'obtenir la factorisation :

$$\mathrm{pgcd}(s - \tilde{s}, N) = p.$$

Une simple faute permet d'extraire la clé secrète.

Les contremesures classiques contre les attaques par fautes consistent à dupliquer les calculs car il est très difficile de faire deux fois la même erreur au même moment du calcul. Dans le cas de la signature RSA, il est aussi possible de vérifier la signature avant de la retourner. Cette méthode est plus efficace que de refaire le calcul car la vérification d'une signature RSA quand l'exposant public est petit, est plus rapide que son calcul.

Attaque sur DES. Sur le schéma de chiffrement par bloc DES [230], Biham et Shamir ont utilisé une attaque par fautes pour retrouver efficacement la clé de chiffrement. Considérons le dernier tour de ce schéma. La valeur qui rentre dans la fonction est la valeur de droite du chiffré R_{16}. Si une erreur est introduite pendant le calcul de la fonction F de l'avant-dernier tour, alors la valeur de gauche $L_{15} = R_{14}$ est inchangée et une erreur apparaît dans R^*_{15} qui est aussi R^*_{16}. Ainsi, nous connaissons la différence $L_{16} \oplus L^*_{16} = F(R_{16}, K_{16}) \oplus F(R^*_{16}, K_{16})$ ainsi que R_{16} et R^*_{16}. La permutation P à la fin de F s'inverse facilement et pour la SBoîte SBi, nous apprenons

$$SBi(K^i_{16} \oplus E(R_{16})^i) \oplus SBi(K^i_{16} \oplus E(R^*_{16})^i),$$

où tout est connu sauf les 6 bits de la sous-clé K_{16} qui rentrent dans SBi. Cette équation est une contrainte sur 4 bits qui permet d'éliminer 1/16 des 64 valeurs possibles pour chacun des 8 mots de 6 bits de la sous-clé K_{16}. Un chiffré correct et un chiffré erroné permettent de réduire l'espace de cette sous-clé à 2^{16} valeurs. En recommençant deux fois en moyenne, nous pouvons déterminer cette sous-clé de manière unique avec une bonne probabilité. Ensuite, une recherche exhaustive sur les 8 derniers bits permet de retrouver toute la clé en utilisant 3 fautes.

Contributions. Avec Denis Réal, Reynald Lercier et Frédéric Valette dans [115], nous avons monté une attaque contre l'algorithme de Montgomery [221, 171], *Montgomery Ladder*, dans le cas de la multiplication par un scalaire sur une courbe elliptique qui permet de retrouver le scalaire.

Cette implémentation est naturellement protégée contre les attaques SPA car aucune instruction ne dépend de bits de la clé. Nous la décrivons dans l'algorithme 6 pour calculer une exponentiation car les calculs de l'exponentiation et de la multiplication par un scalaire sont similaires et nous nous sommes toujours placés dans le cas de l'exponentiation.

Algorithme 6 MONTGOMERY(C, d, N)

Require: 3 entiers $C, d, N \geq 0$.
Ensure: La valeur de $C^d \bmod N$.

1: $N[0] = 1$, $N[1] = C$
2: **for** $i = n$ à 0 **do**
3: $\quad N[\bar{d}_i] = N[\bar{d}_i] \cdot N[d_i] \bmod N$
4: $\quad N[d_i] = N[d_i]^2 \bmod N$
5: **end for**
6: **return** $N[0]$

Soit $d = \sum_{i=0}^{n} d_i 2^i$ l'écriture en binaire de l'exposant privé d. En notant $L_j = \sum_{i=j}^{n} d_i 2^{i-j}$ et $H_j = L_j + 1$, nous avons :

$$L_j = 2L_{j+1} + d_j = L_{j+1} + H_{j+1} + d_j - 1 = 2H_{j+1} + d_j - 2$$

et nous en déduisons :

$$(L_j, H_j) = \begin{cases} (2L_{j+1}, L_{j+1} + H_{j+1}) & \text{si} \quad d_j = 0 \\ (L_{j+1} + H_{j+1}, 2H_{j+1}) & \text{si} \quad d_j = 1 \end{cases}$$

À chaque itération, $N[0]$ contient la valeur C^{L_j} et $N[1]$ la valeur C^{H_j}.

Dans le cas d'une implémentation de l'algorithme de multiplication scalaire sur une courbe elliptique, cette méthode permet de gagner du temps et de la mémoire par rapport à l'algorithme *Double-and-Add*, équivalent de l'algorithme *Square-and-Multiply*, car nous ne sommes pas obligés de calculer et stocker la coordonnée y des points intermédiaires. Les calculs n'ont besoin que de connaître la coordonnée x.

Le corollaire est que les calculs sont identiques si nous travaillons sur la courbe $\mathcal{E}$ d'équation $y^2 = x^3 + ax + b$ sur le corps fini $\mathbb{F}_p$ ou sur une des courbes tordues associées $\tilde{\mathcal{E}}$, de la forme $cy^2 = x^3 + ax + b$ où c est un non-résidu quadratique mod p. Enfin, les abscisses des points de $\mathcal{E}$ et de $\tilde{\mathcal{E}}$ forment une partition du corps fini $\mathbb{F}_p$.

Si nous faisons une erreur au début du calcul dans l'abscisse x du point $P \in \mathcal{E}$, alors avec probabilité $1/2$, le point $\tilde{P}$ se situe sur la courbe $\tilde{\mathcal{E}}$. Si le cardinal du groupe additif engendré par $\tilde{P}$ n'a que des petits facteurs, alors il est possible de calculer des logarithmes discrets et de retrouver k à partir de $k \cdot \tilde{P}$.

Nous avons vérifié que toutes les courbes proposées par le NIST pour ECDSA [227] sauf celle sur 384 bits de sécurité, ne vérifiaient pas cette condition. Sur ces courbes, notre attaque est plus efficace que la méthode générique.

Conclusion

Dans ce mémoire, j'ai présenté certaines attaques algébriques et statistiques en rapport avec mes travaux de recherche dans ce domaine. De nombreux problèmes restent ouverts dans les différents domaines étudiés.

En cryptanalyse de schémas à clé publique, il existe encore des schémas dont la sécurité n'est pas bien assurée. Par exemple, en cryptographie à plusieurs variables, le schéma HFE proposé par Jacques Patarin dans [240] consiste à remplacer le monôme interne du schéma de Matsumoto-Imai [212] par un polynôme de petit degré. Ce schéma a subi plusieurs attaques, mais la complexité des algorithmes pour le résoudre est exponentielle. Il est intéressant de chercher un algorithme en temps polynomial comme c'est le cas pour les instances du schéma de Matsumoto et Imai. De plus, nous ne savons pas actuellement quelle est la sécurité du schéma de signature HFE quand des équations sont enlevées à la clé publique. Les attaques que nous avons décrites sur le schéma SFLASH [97, 96] ne se généralisent pas facilement. Un autre type d'instances particulières sont celles issues du système de signature proposé par Ding *et al.* dans [91] où l'application linéaire de la clé privée, S, n'est pas bijective, mais a un noyau de petite taille. Enfin, le schéma UOV [185] est extrêmement élégant et repose sur l'existence d'un sous-espace totalement isotrope de taille $n/3$ commun à n formes quadratiques en n variables. Avec Gilles Macario-Rat, nous avons montré comment trouver en temps polynomial un espace de taille $n/2$ commun à deux formes quadratiques quand les formes quadratiques sont issues d'un système de Matsumoto-Imai. Cependant, notre méthode ne s'étend pas à des formes quelconques.

Un autre schéma dont la sécurité n'est pas complètement connue est le schéma de chiffrement de McEliece [213], où la clé publique est composée de la matrice de génération d'un code correcteur d'erreur de Goppa masquée en appliquant une application secrète qui est linéaire et bijective d'un côté et une matrice de permutation de l'autre. Récemment, quelques progrès sur certaines instances de code de Goppa ont été réalisés [103] grâce à des méthodes algébriques.

En cryptanalyse de schémas à clé secrète, de nombreux progrès ont été réalisés ces dernières années en ce qui concerne la sécurité des fonctions de hachage. De nouveaux modèles de conception ont été proposés durant la compétition du NIST pour construire une nouvelle fonction de hachage : certaines réutilisent des éléments de la fonction de chiffrement AES, et d'autres mélangent des opérations simples comme l'addition modulo une puissance de 2, les rotations et l'addition modulo 2 sur des mots binaires. De nouvelles techniques d'attaques doivent être imaginées pour analyser la sécurité de ces nouveaux modèles. Quelques attaques ont déjà été développées sur les fonctions utilisant des composants de l'AES. Elles fonctionnent bien quand le nombre de tours est réduit et permettent de trouver efficacement une paire de messages qui suivent un chemin différentiel, même si celui-ci a une probabilité extrêmement faible. Ces attaques ont aussi eu des conséquences imprévisibles sur la sécurité de la fonction de chiffrement AES. En particulier, il a été

montré que l'algorithme de génération de sous-clés de cette fonction a des faiblesses quand la fonction AES est utilisée pour construire une fonction de hachage. Cette faiblesse a ultérieurement été employée pour monter une attaque contre la fonction de chiffrement dans le modèle où l'adversaire peut obtenir le chiffrement de message avec des clés reliées. Plus récemment, une attaque sur la fonction AES utilise un chemin différentiel du type de ceux utilisé dans les attaques sur les fonctions de hachage pour améliorer la complexité en temps d'une attaque par le milieu [100].

Enfin, il est particulièrement intéressant d'étudier la sécurité de la fonction AES sur quelques tours avec peu de couples clairs-chiffrés. De nombreux schémas sont proposés qui utilisent quelques tours de la fonction AES comme la fonction de hachage ECHO, la fonction de code d'authentification de message Alpha-MAC ou le schéma de chiffrement par flot LEX. De même, les attaques par canaux auxiliaires sur la fonction AES permettent souvent d'obtenir des équations portant sur quelques tours. Dans ces attaques, les systèmes à résoudre présentent des caractéristiques intéressantes car ils sont composées d'opérations assez simples et d'applications de boîtes-S. Un algorithme efficace résolvant ce type d'équations serait très utile en pratique.

En ce qui concerne les attaques par canaux auxiliaires, la recherche porte aujourd'hui sur des techniques pour prouver la sécurité contre certains types d'adversaires physiques. Les enjeux dans ce domaine sont de prendre en considération de plus en plus d'adversaires et de décrire et prouver la sécurité de schémas plus efficaces et pratiques que ceux actuellement proposés. La cryptanalyse quant à elle propose de nouveaux modèles de fautes ou de fuites et cherche à les utiliser le plus efficacement possible.

Publications Personnelles

Livres

1. Applied Cryptography and Network Security, 7th International Conference, ACNS'09, Paris-Rocquencourt, France, Juin 2-5, 2009, Proceedings. Springer-Verlag, LNCS 5536.

Articles dans des journaux

1. M. Abdalla, P.-A. Fouque, et D. Pointcheval. *Password-Based Authenticated Key Exchange in the Three-Party Setting*. IEE Proceedings, Volume 153, Issue 1, pp. 27–39, March 2006.
2. C. Bouillaguet, P.-A. Fouque, A. Joux et J. Treger. *A Family of Weak Keys in HFE (and the Corresponding Practical Key-Recovery)*. À paraître dans Journal of Mathematical Cryptology.
3. C. Bouillaguet et P. A. Fouque. *Practical Hash Functions Constructions Resistant to Generic Seconde Preimage Attacks Beyond the Birthday Bound*. À paraître dans Information Processing Letters.
4. E. Andreeva, C. Bouillaguet, O. Dunkelman, P.-A. Fouque, J.J. Hoch, J. Kelsey, A. Shamir et S. Zimmer. *New Second Preimage Attacks on Hash Functions*. À paraître dans Journal of Cryptology.

Articles présentés à des congrès internationaux avec comité de lecture

1. P.A. Fouque et M. Tibouchi. Deterministic encoding and hashing to odd hyperelliptic curves. In *Proceedings of Pairing '10*, volume of *Lecture Notes in Computer Science*, Springer-Verlag, pages 81–91 , 2010.
2. C. Bouillaguet, P.A. Fouque et G. Leurent. Security Analysis of SIMD. In *Proceedings of SAC '10*, volume of *Lecture Notes in Computer Science*, Springer-Verlag, pages – , 2010.
3. C. Bouillaguet, O. Dunkelman, P.A. Fouque et G. Leurent. Attacks on Hash Functions based on Generalized Feistel – Application to Reduced-Round Lesamnta and Shavite-3-512. In *Proceedings of SAC '10*, volume of *Lecture Notes in Computer Science*, Springer-Verlag, pages – , 2010.
4. P.A. Fouque et M. Tibouchi. Estimating the size of the image of deterministic hash functions to elliptic curves. In *Proceedings of Latincrypt '10*, volume of *Lecture Notes in Computer Science*, Springer-Verlag, pages – , 2010.

5. B. Chevalier-Mames, P.A. Fouque, D. Pointcheval, J. Stern et J. Traoré. On Some Incompatible Properties of Voting Schemes. In *Towards Trustworthy Elections*, volume 6000 of *Lecture Notes in Computer Science*, Springer-Verlag, pages 191–199, 2010.
6. C. Bouillaguet, O. Dunkelman, P.A. Fouque et G. Leurent. Another Look at Complementation Properties. In *Proceedings of FSE '10*, volume of *Lecture Notes in Computer Science*, Springer-Verlag, pages 347–364, 2010.
7. P.-A. Fouque, G. Leurent, D. Réal et F. Valette. Practical Electromagnetic Template Attack on HMAC. In *Proceedings of CHES '09*, volume 5747 of *Lecture Notes in Computer Science*, Springer-Verlag, pages 66–80, 2009.
8. P.-A. Fouque, D. Masgana, et F. Valette. Fault Attack on Schnorr Based Identification and Signature Schemes. In *Proceedings of FDTC '09*, IEEE-CS Press, 2009.
9. C. Chevalier, P.-A. Fouque, D. Pointcheval et S. Zimmer. Optimal Randomness Extraction from a Diffie-Hellman Element. In *Proceedings of Eurocrypt '09*, volume 5479 of *Lecture Notes in Computer Science*, Springer-Verlag, pages 572–589 , 2009.
10. C. Bouillaguet et P.-A. Fouque. Analysis of the Collision Resistance of Radiogatún using Algebraic Techniques. In *Proceedings of SAC '08*, volume 5381 of *Lecture Notes in Computer Science*, Springer-Verlag, pages 245–261, 2008.
11. P.-A. Fouque, J. Stern, et S. Zimmer. Cryptanalysis of the Tweaked Versions of SMASH and Reparation. In *Proceedings of SAC '08*, volume 5381 of *Lecture Notes in Computer Science*, Springer-Verlag, pages 136–150, 2008.
12. P.-A. Fouque, D. Réal, F. Valette et M. Drissi. The Carry Leakage on the Randomized Exponent Countermeasure. In *Proceedings of CHES '08*, volume 5154 of *Lecture Notes in Computer Science*, Springer-Verlag, pages 198–213, 2008.
13. P.-A. Fouque, R. Lercier, D. Réal, et F. Valette. Fault Attack on Elliptic Curve with Montgomery Ladder. In *Proceedings of FDTC '08*, IEEE-CS Press, 2008.
14. E. Andreeva, C. Bouillaguet, P.-A. Fouque, J.J. Hoch, J. Kelsey, A. Shamir et S. Zimmer. Second Preimage Attacks on Dithered Hash Functions. In *Proceedings of Eurocrypt '08*, volume 4965 of *Lecture Notes in Computer Science*, Springer-Verlag, pages 270 – 288, 2008.
15. P.-A. Fouque, G. Macario-Rat et J. Stern. Key Recovery on Hidden Monomial Multivariate Schemes. In *Proceedings of Eurocrypt '08*, volume 4965 of *Lecture Notes in Computer Science*, Springer-Verlag, pages 19 – 30, 2008.
16. P.-A. Fouque, G. Martinet, F. Valette, et S. Zimmer. On the Security of the CCM Encryption Mode and of a Slight Variant. In *Proceedings of ACNS '08*, volume of *Lecture Notes in Computer Science*, Springer-Verlag, pages – , 2008.
17. P.-A. Fouque et G. Leurent. Cryptanalysis of a Hash Function Based on Quasi-Cyclic Codes. In *Proceedings of CT RSA '08*, volume of *Lecture Notes in Computer Science*, Springer-Verlag, pages 19 – 35, 2008.
18. P.-A. Fouque, D. Pointcheval, et S. Zimmer. HMAC is a Randomness Extractor and Applications to TLS. In *Proceedings of AsiaCCS '08*, ACM Press, pages 21 – 32, 2008.
19. P.-A. Fouque, G. Macario-Rat, L. Perret et J. Stern. Total Break of the ℓ-IC Signature Scheme. In *Proceedings of PKC '08*, volume 4939 of *Lecture Notes in Computer Science*, Springer-Verlag, pages 1–17, 2008.

20. V. Dubois, P.-A. Fouque, A. Shamir et J. Stern. Practical Cryptanalysis of SFLASH. In *Proceedings of Crypto '07*, volume 4622 of *Lecture Notes in Computer Science*, Springer-Verlag, pages 1 – 12, 2007.
21. P.-A. Fouque, G. Leurent et P.Q. Nguyen. Full Key-Recovery Attacks on HMAC/NMAC-MD4 and NMAC-MD5. In *Proceedings of Crypto '07*, volume 4622 of *Lecture Notes in Computer Science*, Springer-Verlag, pages 13 – 30, 2007.
22. V. Dubois, P.-A. Fouque et J. Stern. Cryptanalysis of SFLASH with Slightly Modified Parameters. In *Proceedings of Eurocrypt '07*, volume 4515 of *Lecture Notes in Computer Science*, Springer-Verlag, pages 264 – 375, 2007.
23. P.-A. Fouque, S. Kunz-Jacques, G. Martinet, F. Muller et F.Valette. Power Attack on Small RSA Public Exponent. In *Proceedings of CHES '06*, volume 4249 of *Lecture Notes in Computer Science*, Springer-Verlag, pages 339 – 353, 2006.
24. P.-A. Fouque, D. Pointcheval, J. Stern et S. Zimmer. Hardness of Distinguishing the MSB or LSB of Secret Keys in Diffie-Hellman Schemes. In *Proceedings of ICALP '06, Part II*, volume 4052 of *Lecture Notes in Computer Science*, Springer-Verlag, pages 240 – 251, 2006.
25. O. Chevassut, P.-A. Fouque, P. Gaudry et D. Pointcheval. The Twist-AUgmented Technique for Key Exchange. In *Proceedings of PKC '06*, volume 3958 of *Lecture Notes in Computer Science*, Springer-Verlag, pages 410 – 426, 2006.
26. E. Levieil et P.-A. Fouque. An Improved LPN Algorithm. In *Proceedings of SCN '06*, volume 4116 of *Lecture Notes in Computer Science*, Springer-Verlag, pages 348 – 359, 2006.
27. M. Abdalla, O. Chevassut, P.-A. Fouque et D. Pointcheval. A Simple Threshold Authenticated Key Exchange from Short Secrets. In *Proceedings of Asiacrypt '05*, volume 3788 of *Lecture Notes in Computer Science*, Springer-Verlag, pages 566 – 584, 2005.
28. P.-A. Fouque, L. Granboulan et J. Stern. Differential Cryptanalysis for Multivariate Schemes. In *Proceedings of Eurocrypt '05*, volume 3494 of *Lecture Notes in Computer Science*, Springer-Verlag, pages 341–353, 2005.
29. M. Abdalla, P.-A. Fouque, et D. Pointcheval. Password-Based Authenticated Key Exchange in the Three-Party Setting. In *Proceedings of PKC '05*, volume 3386 of *Lecture Notes in Computer Science*, Springer-Verlag, pages 65–84, 2005.
30. P.-A. Fouque, F. Muller, G. Poupard et F. Valette. Defeating Countermeasures Based on Randomized BSD Representations. In *Proceedings of CHES '04*, volume 3156 of *Lecture Notes in Computer Science*, Springer-Verlag, pages 312–327, 2004.
31. P.-A. Fouque, A. Joux et G. Poupard. Blockwise Adversarial Model for On-line Ciphers and Symmetric Encryption Schemes. In *Proceedings of SAC '04*, volume 3357 of *Lecture Notes in Computer Science*, Springer-Verlag, pages 212–226, 2004.
32. P.-A. Fouque, N. Howgrave-Graham, G. Martinet et G. Poupard. The Insecurity of Esign in Practical Implemenrations. In *Proceedings of Asiacrypt '03*, volume 2894 of *Lecture Notes in Computer Science*, Springer-Verlag, pages 492–506, 2003.
33. P.-A. Fouque, G. Martinet et G. Poupard. Attacking Unbalanced RSA-CRT Using SPA. In *Proceedings of CHES '03*, volume 2779 of *Lecture Notes in Computer Science*, Springer-Verlag, pages 254–268, 2003.
34. P.-A. Fouque et F. Valette. The Doubling Attack - *Why Upwards is Better than Downwards*. In *Proceedings of CHES '03*, volume 2779 of *Lecture Notes in Computer Science*, Springer-Verlag, pages 269–280, 2003.

35. P.-A. Fouque et G. Poupard. On the Security of RDSA. In *Proceedings of Eurocrypt '03*, volume 2656 of *Lecture Notes in Computer Science*, Springer-Verlag, pages 462–476, 2003.
36. P.-A. Fouque, G. Martinet et G. Poupard. Practical Symmetric On-Line Encryption. In *Proceedings of FSE '03*, volume 2887 of *Lecture Notes in Computer Science*, Springer-Verlag, pages 462–476, 2003.
37. P.-A. Fouque, A. Joux, G. Martinet et F. Valette. Authenticated On-Line Encryption. In *Proceedings of SAC '03*, volume 3006 of *Lecture Notes in Computer Science*, Springer-Verlag, pages 145–159, 2003.
38. P.-A. Fouque, J. Stern, et J.-G. Wackers. CryptoComputing with Rationals. In *Proceedings of FC '02*, volume 2357 of *Lecture Notes in Computer Science*, Springer-Verlag, pages 136–146, 2002.
39. P.-A. Fouque et J. Stern. Fully Distributed Threshold RSA under Standard Assumptions. In *Proceedings of Asiacrypt '01*, volume 2248 of *Lecture Notes in Computer Science*, Springer-Verlag, pages 310–330, 2001.
40. P.-A. Fouque et D. Pointcheval. Threshold Cryptosystems Secure against Chosen-Ciphertext Attacks. In *Proceedings of Asiacrypt '01*, volume 2248 of *Lecture Notes in Computer Science*, Springer-Verlag, pages 351–368, 2001.
41. O. Baudron, P.-A. Fouque, D. Pointcheval, G. Poupard et J. Stern. Practical multi-candidate election system. In *Proceedings of PODC '01*, ACM Press, pages 274–283, 2001.
42. P.-A. Fouque et J. Stern. One Round Threshold Discrete-Log Key Generation without Private Channels. In *Proceedings of PKC '01*, volume 1992 of *Lecture Notes in Computer Science*, Springer-Verlag, pages 300–316, 2001.
43. P.-A. Fouque, G. Poupard et J. Stern. Sharing Decryption in the Context of Voting or Lotteries. In *Proceedings of FC '00*, volume 1962 of *Lecture Notes in Computer Science*, Springer-Verlag, pages 90–104, 2000.

Attaques algébriques

Annexe A :

Differential Cryptanalysis for Multivariate Schemes, EUROCRYPT 2005
Pierre-Alain Fouque, Louis Granboulan et Jacques Stern
Cet article présente une attaque différentielle sur le schéma de chiffrement PMI et introduit les attaques différentielles contre les schémas où la clé publique est formée d'un système d'équations quadratiques. Cet outil est très efficace car il permet de trouver des relations bilinéaires entre le clair et le chiffré. Dans cette attaque, nous exploitons le fait que pour les schémas issus du schéma de Matsumoto-Imai, le noyau de la différentielle a une forme particulière.

Annexe B :

Practical Cryptanalysis of SFLASH, CRYPT0 2007
Vivien Dubois, Pierre-Alain Fouque, Adi Shamir et Jacques Stern
Cet article présente l'attaque du schéma de signature SFLASH. Dans ce schéma de signature, des équations quadratiques ont été enlevées à la clé publique d'un système de Matsumoto-Imai pour éviter l'attaque de Patarin. Nous avons montré qu'à partir de la clé publique, nous pouvons retrouver des équations équivalentes à celle de départ sur lesquelles l'attaque de Patarin s'applique.

Annexe C :

Key Recovery on Hidden Monomial Multivariate Schemes, EUROCRYPT 2008
Pierre-Alain Fouque, Gilles Macario-Rat et Jacques Stern
Cet article présente une attaque pour retrouver la clé secrète du schéma SFLASH. L'attaque de l'article B permet de signer n'importe quel message en retrouvant une clé publique complète sur laquelle l'attaque de Patarin s'applique. Ici, nous montrons qu'il est possible de retrouver la clé secrète en utilisant des outils algébriques. De plus, nous étendons l'attaque précédente pour des monômes de degré plus élevé, c'est-à-dire de la forme $X^{1+q^{\theta_1}+\ldots+q^{\theta_d}}$.

Annexe D :

Probabilistic Algorithms for the Equivalence of Quadratic Maps,
Charles Bouillaguet, Pierre-Alain Fouque et Amandine Véber
Cet article présente et analyse la complexité de deux algorithmes pour résoudre le problème d'Isomorphisme de Polynômes (IP) : étant donné A *et* B *deux systèmes de* n *équations quadratiques en* n *variables, trouver* S *et* T *deux matrices inversibles telles que* $B = T \circ A \circ S$. *La complexité de ces algorithmes est en* $\tilde{O}(q^{2n/3})$ *pour le premier et* $\tilde{O}(q^{n/2})$ *pour le second. Ces algorithmes reposent sur un paradoxe des anniversaires pour trouver efficacement des couples* $(x, S(x))$.

Annexe A

Differential Cryptanalysis for Multivariate Schemes

EUROCRYPT 2005
[110] avec Louis Granboulan et Jacques Stern

Abstract : *In this paper we propose a novel cryptanalytic method against multivariate schemes, which adapts differential cryptanalysis to this setting. In multivariate quadratic systems, the differential of the public key is a linear map and has invariants such as the dimension of the kernel. Using linear algebra, the study of this invariant can be used to gain information on the secret key. We successfully apply this new method to break the original Matsumoto-Imai cryptosystem using properties of the differential, thus providing an alternative attack against this scheme besides the attack devised by Patarin. Next, we present an attack against a randomised variant of the Matsumoto-Imai cryptosystem, called PMI. This scheme has recently been proposed by Ding, and according to the author, it resists all previously known attacks. We believe that differential cryptanalysis is a general and powerful method that can give additional insight on most multivariate schemes proposed so far.*

A.1 Introduction

The design of efficient and secure cryptosystems is a hard task. Many alternatives to the traditional public key cryptosystems (RSA, ElGamal) have been proposed so far but few of them are considered secure. An interesting line of research is based on multivariate quadratic polynomials over a finite field. This line of research has been initiated by Matsumoto and Imai [212]. These systems are attractive since the underlying problem is known to be NP-complete and the decryption algorithm is more efficient than the RSA algorithm.

The original cryptosystem of Matsumoto and Imai (MI or C^*) has been broken by Patarin [238] who has also proposed various techniques that protect against this attack [239, 240]. A generalisation of MI, called Hidden Field Equations (HFE) [240], has higher security, but it has nevertheless been broken by Kipnis and Shamir [186]. More efficient attacks were proposed by Courtois *et al.* in [80, 81] and culminated with Faugère and Joux attack and the use of Gröbner bases in [102].

Variants of the original MI scheme remain interesting because they achieve better performance than variants of HFE. The main variants of MI that resist the attack by Patarin are on one hand, the Minus method which consists in discarding a few polynomials in the public key, and on the other hand the Minus-Plus method, which proposes to discard some polynomials and to add a few variables. These methods use *external* perturbation of the MI scheme, since variables are removed after the application of the exponentiation function.

Recently, Ding [90] proposed a new variant of the MI cryptosystem using some *internal* perturbation, which occurs before applying the exponentiation function. He quickly analyses its proposal against all known attacks on multivariate schemes, and claims that it is immune against such attacks. The new scheme is nearly as efficient as the original MI and the author gives some arguments in order to show that its scheme, called Perturbated MI (PMI), is a more secure extension than the MI Minus and MI Minus-Plus method.

A.1.1 Our Results

In this paper, we describe a new technique which is extremely powerful and that could presumably be used to break other multivariate schemes. In order to illustrate the power and generality of this method, we first propose a new attack on the original MI scheme and next describe how it can be used to mount an attack against the PMI cryptosystem.

The key point of our attack is that in the case of quadratic polynomials, the differential of the public key is a linear map and its kernel or its rank can be analysed to get some information on the secret key. For example, in the PMI scheme, we show that the dimension of the kernel can be used to identify elements that cancel the perturbation. In fact, we design a one-sided error recogniser for the language of elements that are not in the kernel of the perturbation. From this test algorithm, we design two algorithms to reconstruct the kernel. These algorithms are of independent interest. With the first method, the complexity of the attack is a precomputation of order $O(nq^{3r} + n^6 q^r)$, which can be upperbounded by 2^{49} with the proposed parameters in [90], and $O(n^3 \times q^r \times q^{\gcd(\ell,n)})$, which is of order 2^{36} binary operations. Finally, this attack works for scheme over finite fields of characteristic 2 which are the main structure for efficiency reasons and for MI and PMI this is always the case as we will see. In the case of the original MI cryptosystem, we use elements in the kernel of the transpose of the differential in order to propose a new attack. We actually prove a bilinear relation between the ciphertext and the kernel vector. Thus, the kernel allows to recover the plaintext by solving a linear system.

A.1.2 Related Works

Differentials have already been successfully applied to break multivariate schemes such as the Minus transformation of the original Matsumoto-Imai, or the SFLASH signature scheme or the "2R" scheme proposed by Patarin [239, 242, 137, 297, 30]. Our work gives a better insight by bringing a systematic use of the geometric properties of the differential.

A.1.3 Organisation of the paper

In section 2 of this paper, we describe the MI and PMI cryptosystems. Then, in section 3 we recall Patarin's attack on the original MI scheme. Next in section 4, we describe our attack on the PMI scheme and some experimental results. Finally, in section 5, we show a new attack on the original MI scheme.

A.2 Description of the MI and PMI schemes

A.2.1 The Matsumoto-Imai cryptosystem

This scheme is based on the following fact : over the finite field $\mathbb{F}_{q^n}$, the function $F : x \mapsto x^{q^\ell+1}$ is a permutation, when $\gcd(q^\ell + 1, q^n - 1) = 1$. Therefore, we can fix q to be a power of 2 so that $\mathbb{F}_{q^n}$ is of characteristic two [1]. Its inverse is $x \mapsto x^h$ where h is the inverse of $q^\ell + 1$ in $\mathbb{Z}_{q^n-1}$. Therefore, for any isomorphism π from the vector space of $\mathbb{F}_{q^n}$ to the n-dimensional vector space $(\mathbb{F}_q)^n$, the function $\boldsymbol{F} = \pi \circ F \circ \pi^{-1}$ is a bijective system of multivariate quadratic functions since F can be viewed as the product of two linear maps $x \mapsto x^{q^\ell}$ and $x \mapsto x$.

The scheme described by Matsumoto and Imai in 1988 [212] generates $\boldsymbol{S}$ and $\boldsymbol{T}$, two secret affine bijections of $(\mathbb{F}_q)^n$ to mask the system $\boldsymbol{F}$. The system $\boldsymbol{E} = \boldsymbol{T} \circ \boldsymbol{F} \circ \boldsymbol{S}$ is also a system of multivariate quadratic equations and represents the public key. Patarin showed in 1995 [238] that the public key has a special form which allows to invert the function.

A.2.2 The PMI cryptosystem

Recently at PKC '04, Ding proposed a randomised variant of MI, called PMI [90]. Let $\boldsymbol{R} : (\mathbb{F}_q)^n \to (\mathbb{F}_q)^r$ a secret linear function of small rank ($r \ll n$) and $\boldsymbol{H}$ a secret quadratic system composed of n quadratic equations over r variables. The PMI public key is the system $\boldsymbol{E'}$ defined by $\boldsymbol{E'} = \boldsymbol{T} \circ (\boldsymbol{F} + \boldsymbol{H} \circ \boldsymbol{R}) \circ \boldsymbol{S}$. The public key can also be written as $\boldsymbol{E'} = \boldsymbol{T} \circ \boldsymbol{F} \circ \boldsymbol{S} + \boldsymbol{T} \circ \boldsymbol{H} \circ \boldsymbol{R} \circ \boldsymbol{S}$ due to the linearity of $\boldsymbol{T}$. Consequently, the PMI scheme can be seen as the MI scheme $\boldsymbol{E}$ plus a random-looking quadratic term $\boldsymbol{T} \circ \boldsymbol{H} \circ \boldsymbol{R} \circ \boldsymbol{S}$. Since there is no trapdoor to invert $\boldsymbol{H}$ or to separate the MI term and the random term, we need to store all the inputs and the outputs of the $\boldsymbol{H}$ function. Let P be the set of points which consist of pairs $(\boldsymbol{\lambda}, \boldsymbol{\mu})$, where $\boldsymbol{\lambda}$ is a point that belongs to the image of $\boldsymbol{H}$, and $\boldsymbol{\mu}$ is the set of pre-images of $\boldsymbol{\lambda}$ under $\boldsymbol{H}$. The set P contains q^r points. The secret key includes the set of linear functions $\boldsymbol{R}$, the set P, and the two affine bijections $\boldsymbol{S}$ and $\boldsymbol{T}$.

The secret key allows to invert $\boldsymbol{E'}$ if one can make exhaustive search over the q^r values of P and so r must be small. More precisely, given a ciphertext $\boldsymbol{y}$, the decryption process inverses the affine bijection $\boldsymbol{T}$ and recovers $\boldsymbol{y'}$. Then, all elements $(\boldsymbol{\lambda}, \boldsymbol{\mu})$ in P can be tried one-by-one and $\boldsymbol{y'_\lambda} = \boldsymbol{F}^{-1}(\boldsymbol{y'} + \boldsymbol{\lambda})$ is computed. Next, if $\boldsymbol{H}(\boldsymbol{y'_\lambda})$ is not equal to $\boldsymbol{\mu}$, we try the next point in P, otherwise, we compute $\boldsymbol{x_\lambda}$ by $\boldsymbol{S}^{-1}(\boldsymbol{y'_\lambda})$. If we have only one solution, we get the plaintext, otherwise, we use some added redundancy in the plaintext in order to uniquely recover it.

In his description of PMI [90], Ding analyses all known attack such as algebraic attacks of Patarin [238], Kipnis and Shamir [186], or XL attacks [82] and the attack on MI Minus of Patarin, Goubin and Courtois [242].

He also proposes a practical implementation with $q = 2$, $n = 136$, $r = 6$ and $F(x) = x^{2^{5\times8}+1}$. He claims that the security level for this choice of parameters is 2^{136}. The value ℓ has been chosen with a special form, such that $\gcd(2^n - 1, 2^\ell - 1) = 2^{\gcd(n,\ell)} - 1 = 2^{\gcd(136,5\times8)} - 1 = 2^8 - 1$. This special form allows to perform more efficient encryption and decryption using lookup tables for the multiplications in the finite field.

In this paper, we apply differential cryptanalysis to the PMI scheme, and we show that the special form of the exponent in the practical system proposed by Ding allows more efficient attack than the attack in the generic case where $\gcd(\ell, n) = 1$.

1. Indeed, if q is odd, we have $\gcd(q^\ell + 1, q^n - 1) \geq 2$ and since q is a prime power, it is always a power of 2 and the characteristic of $\mathbb{F}_{q^n}$ is 2

A.3 Patarin's Attack on the MI cryptosystem

Our attack against the PMI cryptosystem is a probabilistic reduction to Patarin's attack on the MI scheme. Therefore, prior the description of our attack, we recall Patarin's attack. While we also propose an alternative attack to the MI scheme in section A.5, we present Patarin attack since it is easier to understand. Both his attack and our attack do not recover the secret key but finds a linear system which can be solved to recover the plaintext corresponding to a given ciphertext.

Let $\boldsymbol{x} \in (\mathbb{F}_q)^n$ a plaintext and $\boldsymbol{y} \in (\mathbb{F}_q)^n$ the corresponding ciphertext. The main idea of Patarin attack is to find several bilinear relations in the $\boldsymbol{x}$ and $\boldsymbol{y}$ coordinates. Using plaintext/ciphertext pairs $(\boldsymbol{x}, \boldsymbol{y})$, it is possible to recover the coefficients of the relations by solving a linear system. Finally, knowing these coefficients and a given ciphertext, it is possible to decrypt $\boldsymbol{y}$ by solving a linear system.

Let us define $a = \pi^{-1}(\boldsymbol{S}(\boldsymbol{x}))$ and $b = \pi^{-1}(\boldsymbol{T}^{-1}(\boldsymbol{y}))$. Consequently, $F(a) = b$ or $b = a^{q^\ell+1}$. By raising each member of the last equation to the power $q^\ell - 1$ and by multiplying each one by ab, we get

$$ab^{q^\ell} = a^{q^{2\ell}}b \tag{A.1}$$

which holds over the finite field $\mathbb{F}_{q^n}$. We can rewrite this equation by $B(a, b) = 0$ where $B(a, b) = a \cdot b^{q^\ell} - a^{q^{2\ell}} \cdot b$. If we represent equation (A.1) in $(\mathbb{F}_q)^n$, we get n bilinear equations in the n coordinates of $\boldsymbol{a}$ and of $\boldsymbol{b}$. As $\boldsymbol{a}$ and $\boldsymbol{b}$ are affine transformations of $\boldsymbol{x}$ and $\boldsymbol{y}$ via the secret affine bijections $\boldsymbol{S}$ and $\boldsymbol{T}$, the n bilinear expressions in $\boldsymbol{a}$ and $\boldsymbol{b}$, may also be written as n bilinear expressions in $\boldsymbol{x}$ and $\boldsymbol{y}$. Each expression can be written as $\sum_{i=1}^{n}\sum_{j=1}^{n}\beta_{i,j}x_iy_j + \sum_{i=1}^{n}\beta_{i,0}x_i + \sum_{j=1}^{n}\beta_{0,j}y_j + \beta_{0,0} = 0$.

For each plaintext/ciphertext pair $(\boldsymbol{x}, \boldsymbol{y})$, the equation above, where all the $\beta_{i,j}$ are the $(n+1)^2$ unknowns, has at least the n solutions described by the n bilinear expressions deduced from equation (A.1). Therefore, using $\mathcal{O}((n+1)^2)$ plaintext/ciphertext pairs, solving the resulting system of $\mathcal{O}((n+1)^2)$ equations in the $(n+1)^2$ unknowns $\beta_{i,j}$ will recover the n bilinear expressions.

Finally, given a ciphertext $\boldsymbol{y}$ to decrypt, these n equations will give us n linear equations in the coefficients of $\boldsymbol{x}$. Unfortunately, all these equations are not independent. The solutions of this system correspond to the solutions of (A.1). There are $q^{\gcd(n,\ell)}$ such solutions, as shown by Patarin : let us consider the equation (A.1) where the unknown is a. A ciphertext $\boldsymbol{y}$ fixes a unique b value. One solution is $a = 0$. If $a \neq 0$ (and so $b \neq 0$) the equation can be written as

$$a^{q^{2\ell}-1} = b^{q^\ell-1}$$

We can write $q^{2\ell}-1$ as $(q^\ell+1)(q^\ell-1)$ and take the inverse of $q^\ell+1$ modulo q^n-1 since by assumption F is a permutation. Consequently, the equation becomes $a^{q^\ell-1} = b^{h\times(q^\ell-1)} = b'$ where h is the inverse of $q^\ell+1$. This last equation has exactly $\gcd(q^\ell-1, q^n-1) = q^{\gcd(\ell,n)}-1$ solutions as shown in appendix A.6 since the right solution is one solution.

As a consequence, the solution that we are looking for is a particular vector of the kernel of some system related to the original system, and the second member of the equation [73, p. 59]. Therefore, we compute the kernel of the system matrix which is of dimension $\gcd(n, \ell)$. Next, we perform an exhaustive search in $q^{\gcd(n,\ell)} - 1$ coefficients of the kernel vector, in order to recover the correct value $\boldsymbol{x}$.

In section A.5, we propose a new differential attack on the MI scheme by studying the kernel of the transpose of the differential of the public key. We show that there exist n bilinear forms between a ciphertext $\boldsymbol{E(k)}$ and the vector $\boldsymbol{f_k}$ that generates the kernel

of the transpose of the differential, which is of dimension 1 if $\gcd(\ell, n) = 1$. Then, given a ciphertext, we are able to reconstruct the vector $\boldsymbol{f_k}$ since the n bilinear forms are independent as there is a unique solution for the n bilinear forms. Finally, since the vector $\boldsymbol{f_k}$ is in the kernel of the transpose of the differential and that this map is linear in $\boldsymbol{k}$, we can solve n linear equations in the $\boldsymbol{k}$ variables of n coordinates. We refer the reader to section A.5 for details.

A.4 Cryptanalysis of the PMI cryptosystem

A.4.1 Overview of the attack

Let us recall the notations : $\boldsymbol{F}$ is the system of quadratic equations corresponding to the internal function of the MI cryptosystem, $\boldsymbol{E} = \boldsymbol{T} \circ \boldsymbol{F} \circ \boldsymbol{S}$ the public key of MI, and $\boldsymbol{E'} = \boldsymbol{E} + \boldsymbol{T} \circ \boldsymbol{H} \circ \boldsymbol{R} \circ \boldsymbol{S}$ the public key of PMI.

Our attack is based on the following remark : the PMI scheme is a noisy MI cryptosystem. We find the linear space $\mathcal{K}$ that cancels the noise, and apply an attack of MI to the restriction of PMI to this linear space.

More precisely, we define the linear space $\mathcal{K}$ as follows : it is the kernel of the linear part of the affine function $\boldsymbol{R} \circ \boldsymbol{S}$. The space $\mathcal{K}$ is of dimension $\dim(\ker \boldsymbol{R}) = n - r$ because $\boldsymbol{S}$ is a bijection and $\operatorname{rank}(\boldsymbol{R}) = r$. If we are able to compute $\mathcal{K}$, then we can apply the attacks against MI (either Patarin's attack or our attack described in section A.5) to the PMI cryptosystem restrict to elements of one of the q^r affine spaces that are parallel to $\mathcal{K}$. When restrict to one of these affine spaces, the public key of PMI is exactly $\boldsymbol{E}$ translated by a constant. The attack of PMI amounts to q^r attacks against MI (this is feasible because q^r must be of moderate size to allow fast decryption). A ciphertext is decrypted by applying the attack to the affine space that contains its corresponding plaintext.

In order to recover the space $\mathcal{K}$, we devise an efficient test algorithm that can spot that a given vector $\boldsymbol{k}$ does not belong to $\mathcal{K}$. The information used in this test is the dimension of the kernel of the linear part of the differential of the public key.

A.4.2 The dimension of the kernel of the differential

For any function $\boldsymbol{G} : (\mathbb{F}_q)^n \to (\mathbb{F}_q)^m$, let us consider its differential $\boldsymbol{dG_k}(x) = \boldsymbol{G}(\boldsymbol{x} + \boldsymbol{k}) - \boldsymbol{G}(\boldsymbol{x})$. Because $\boldsymbol{G}$ is a quadratic function, its differential is an affine function. Let us consider $\boldsymbol{L_{G,k}}(x) = \boldsymbol{dG_k}(x) - \boldsymbol{dG_k}(0)$ the linear part of the differential. In fact, it is a bilinear function that can also be defined by $\boldsymbol{L_{G,k}}(x) = \boldsymbol{B_G}(\boldsymbol{x}, \boldsymbol{k}) = \boldsymbol{G}(\boldsymbol{x} + \boldsymbol{k}) - \boldsymbol{G}(\boldsymbol{x}) - \boldsymbol{G}(\boldsymbol{k}) + \boldsymbol{G}(\boldsymbol{0})$, and is also called the polar form. We are interested in $\dim(\ker \boldsymbol{L_{G,k}})$ when $\boldsymbol{G}$ is the public key of the cryptosystem.

Property A.1. *Let $\boldsymbol{k}$ and $\boldsymbol{k'}$ be elements of $(\mathbb{F}_q)^n$, and $\boldsymbol{G}$ and $\boldsymbol{G'}$ be systems of quadratic equations, and $\boldsymbol{S}$ and $\boldsymbol{T}$ be affine bijections. The following properties hold :* $\boldsymbol{L_{G,k+k'}} = \boldsymbol{L_{G,k}} + \boldsymbol{L_{G,k'}}$, $\boldsymbol{L_{G+G',k}} = \boldsymbol{L_{G,k}} + \boldsymbol{L_{G',k}}$, $\boldsymbol{L_{T \circ G \circ S,k}} = \boldsymbol{T} \circ \boldsymbol{L_{G,S(k)}} \circ \boldsymbol{S} + \boldsymbol{T} \circ \boldsymbol{G} \circ \boldsymbol{S}(\boldsymbol{0}) - \boldsymbol{T} \circ \boldsymbol{G}(\boldsymbol{0})$, *and* $\boldsymbol{L_{G,0}} = \boldsymbol{0}$. ▽

Lemme A.2. *If $\boldsymbol{E}$ is the public key of a MI system over $\mathbb{F}_q$ of characteristic 2, of dimension n and exponent $q^\ell + 1$, then* $\dim(\ker \boldsymbol{L_{E,k}}) = \gcd(\ell, n)$. ▽

First, $\dim \ker(\boldsymbol{L_{E,k}}) = \dim \ker(\boldsymbol{L_{F,k}})$, because $\boldsymbol{T}$ and $\boldsymbol{S}$ are bijections.

Let us define $\boldsymbol{x} = \pi(x)$ and $\boldsymbol{k} = \pi(k)$. If $\boldsymbol{F}$ is the internal function of the MI cryptosystem, then $\boldsymbol{B_F}(\boldsymbol{x}, \boldsymbol{k})$ is equal to $\pi(x^{q^\ell} \cdot k + x \cdot k^{q^\ell})$. A vector $\boldsymbol{x} \neq \boldsymbol{0}$ of $(\mathbb{F}_q)^n$ is in the

kernel of $\boldsymbol{L_{F,k}}$ if and only if $x^{q^\ell} \cdot k + x \cdot k^{q^\ell} = 0$. This last equation can be written as $x^{q^\ell+1} \cdot \left(\frac{k}{x} + \left(\frac{k}{x}\right)^{q^\ell}\right) = 0$.

Since $x \neq 0$, if we denote k/x by X, then the previous equation is $X + X^{q^\ell} = 0$ in the finite field $\mathbb{F}_{q^n}$. If $X \neq 0$ ($k \neq 0$), then the equation becomes $X^{q^\ell-1} = 1$ in a finite field of characteristic 2. Since $X = 1$ is solution, there is at least one solution. As a consequence, there are $q^{\gcd(\ell,n)} - 1$ solutions according to the results in appendix A.6, and therefore $\dim(\ker \boldsymbol{L_{E,k}}) = \gcd(\ell, n)$.

Note that $X = 1$ is always a solution, that means $x = k$, and therefore $\boldsymbol{k}$ is always in the kernel. There are no other solutions when $\gcd(\ell, n) = 1$.

Lemme A.3. *If $\boldsymbol{E'}$ is the public key of the PMI cryptosystem and $\boldsymbol{k} \in \mathcal{K}$, then* $\dim(\ker \boldsymbol{L_{E',k}}) = \gcd(\ell, n)$. ▽

We prove that if $\boldsymbol{k} \in \mathcal{K}$, then $\boldsymbol{L_{E',k}} = \boldsymbol{L_{E,k}}$. First we notice that $\boldsymbol{k} \in \mathcal{K}$ is equivalent to $\boldsymbol{R} \circ \boldsymbol{S}(\boldsymbol{k}) = \boldsymbol{R} \circ \boldsymbol{S}(\boldsymbol{0})$.

Then we compute $\boldsymbol{L_{E',k}}(\boldsymbol{x}) - \boldsymbol{L_{E,k}}(\boldsymbol{x}) = \boldsymbol{L_{T\circ H\circ R\circ S,k}}(\boldsymbol{x}) = \boldsymbol{T} \circ \boldsymbol{H} \circ \boldsymbol{R} \circ \boldsymbol{S}(\boldsymbol{x}+\boldsymbol{k}) - \boldsymbol{T} \circ \boldsymbol{H} \circ \boldsymbol{R} \circ \boldsymbol{S}(\boldsymbol{x}) - \boldsymbol{T} \circ \boldsymbol{H} \circ \boldsymbol{R} \circ \boldsymbol{S}(\boldsymbol{k}) + \boldsymbol{T} \circ \boldsymbol{H} \circ \boldsymbol{R} \circ \boldsymbol{S}(\boldsymbol{0})$, therefore $\boldsymbol{T}^{-1}(\boldsymbol{L_{E',k}}(\boldsymbol{x}) - \boldsymbol{L_{E,k}}(\boldsymbol{x})) = \boldsymbol{H}(\boldsymbol{R} \circ \boldsymbol{S}(\boldsymbol{x} + \boldsymbol{k})) - \boldsymbol{H}(\boldsymbol{R} \circ \boldsymbol{S}(\boldsymbol{x})) - \boldsymbol{H}(\boldsymbol{R} \circ \boldsymbol{S}(\boldsymbol{k})) + \boldsymbol{H}(\boldsymbol{R} \circ \boldsymbol{S}(\boldsymbol{0})) = \boldsymbol{0}$, which means that $\boldsymbol{T}^{-1} \circ \boldsymbol{L_{E',k}} = \boldsymbol{T}^{-1} \circ \boldsymbol{L_{E,k}}$. Therefore $\dim(\ker \boldsymbol{L_{E',k}}) = \dim(\ker(\boldsymbol{T}^{-1} \circ \boldsymbol{L_{E',k}})) = \dim(\ker(\boldsymbol{T}^{-1} \circ \boldsymbol{L_{E,k}})) = \dim(\ker \boldsymbol{L_{E,k}})$.

Lemme A.4. *If $\boldsymbol{E'}$ is the public key of the PMI cryptosystem and $\boldsymbol{k} \notin \mathcal{K}$, then often* $\dim(\ker \boldsymbol{L_{E',k}}) \neq \gcd(\ell, n)$. ▽

As before, $\boldsymbol{L_{E',k}}$ is the sum of $\boldsymbol{L_{E,k}}$ and $\boldsymbol{L_{T\circ H\circ R\circ S,k}}$. However, when, $\boldsymbol{k} \notin \mathcal{K}$, the second linear application is not null. The argument behind lemma A.4 is that $\boldsymbol{L_{T\circ H\circ R\circ S,k}}$ is a random-looking linear application, and therefore the dimension of the kernel of the sum $\boldsymbol{L_{E',k}}$ follows the distribution of the dimension of the kernel of random linear maps.

In fact, it is slightly more complicated, because $\boldsymbol{k}$ is always in the kernel of $\boldsymbol{L_{T\circ H\circ R\circ S,k}}$, and therefore also in the kernel of $\boldsymbol{L_{E',k}}$, whose dimension then is at least 1. Moreover, if $\gcd(\ell, n) > r$, then there are $\gcd(\ell, n) - r$ additional vectors in the kernel of $\boldsymbol{L_{E',k}}$, because $\ker(\boldsymbol{L_{E,k}})$ of dimension $\gcd(\ell, n)$ and $\ker(\boldsymbol{R} \circ \boldsymbol{S})$ of dimension $n - r$ in a space of dimension n have an intersection of dimension at least $\gcd(\ell, n) - r$. In the case of the practical scheme proposed by Ding where $\gcd(\ell, n) = 8$ and $r = 6$, we can deduce that $\dim(\ker(\boldsymbol{L_{E',k}})) \geq 3$.

Lemma A.4 can be verified experimentally, as shown in table A.1.

TABLE A.1 – Experimental results for the probability distribution of $\dim(\ker(\boldsymbol{L_{E',k}}))$.

$\ell = 41$, $n = 137$ and $r = 6$

dimension	$\boldsymbol{k} \in \mathcal{K}$	$\boldsymbol{k} \notin \mathcal{K}$
1	1	≈ 0.59
> 1	0	≈ 0.41

$\ell = 40$, $n = 136$ and $r = 6$

dimension	$\boldsymbol{k} \in \mathcal{K}$	$\boldsymbol{k} \notin \mathcal{K}$
3	0	≈ 0.686
4	0	≈ 0.290
5	0	≈ 0.023
6	0	$\approx 5.10^{-4}$
7	0	$\approx 2.10^{-6}$
8	1	≈ 0
> 8	0	≈ 0

As a consequence of the lemmas, we get the following corollary.

Corollaire A.5. *If $\boldsymbol{E'}$ is the public key of the PMI cryptosystem and if* $\dim(\ker \boldsymbol{L}_{\boldsymbol{E'},\boldsymbol{k}}) \neq \gcd(\ell, n)$*, then* $\boldsymbol{k} \notin \mathcal{K}$*.* ▽

In conclusion, we have now an efficient test to know if a vector is not in $\mathcal{K}$. We define $T(\boldsymbol{k})$ to be this test : $T(\boldsymbol{k}) = 1$ if $\dim(\ker \boldsymbol{L}_{\boldsymbol{E'},\boldsymbol{k}}) \neq \gcd(\ell, n)$, meaning that $\boldsymbol{k}$ is not in $\mathcal{K}$ with probability one, and $T(\boldsymbol{k}) = 0$ if $\dim(\ker \boldsymbol{L}_{\boldsymbol{E'},\boldsymbol{k}}) = \gcd(\ell, n)$, meaning that $\boldsymbol{k}$ can be in $\mathcal{K}$ or not. Now, we must transform this test into an algorithm for recovering $\mathcal{K}$.

A.4.3 Recovering $\mathcal{K}$

We are looking for $\dim(\mathcal{K})$ independent vectors that generate $\mathcal{K}$. Let us define $\alpha = \Pr[T(\boldsymbol{k}) = 0]$ and $\beta = \Pr[\boldsymbol{k} \in \mathcal{K}] = q^{-r}$. The following table summarises the distribution of the values of T applied to a random $\boldsymbol{k}$.

	$\boldsymbol{k} \in \mathcal{K}$	$\boldsymbol{k} \notin \mathcal{K}$	
$T(\boldsymbol{k}) = 0$	β	$\alpha - \beta$	α
$T(\boldsymbol{k}) = 1$	0	$1 - \alpha$	$1 - \alpha$
	β	$1 - \beta$	

In the case where $\gcd(\ell, n) = 8$ we have $\alpha - \beta \ll \beta$ and therefore the test T has almost no false positives. In the case where $\gcd(\ell, n) = 1$ we have $\beta \ll \alpha$ and therefore the test T cannot give direct proof of membership of $\mathcal{K}$. A specific algorithm to recover $\mathcal{K}$ is needed.

The property we use is the linearity of $\mathcal{K}$: if $\boldsymbol{k}, \boldsymbol{k'} \in \mathcal{K}$, then $\boldsymbol{k}+\boldsymbol{k'} \in \mathcal{K}$. Two algorithms are described below. The first algorithm uses a statistical bias for $T(\boldsymbol{k} + \boldsymbol{k'})$. The second algorithm searches some large clique in a graph. A concrete attack of the PMI cryptosystem will use a mix of both techniques.

Technique 1.

The key idea is : if for many different $\boldsymbol{k'} \in \mathcal{K}$, $\boldsymbol{k} + \boldsymbol{k'}$ is in $\mathcal{K}$, then $\boldsymbol{k}$ is always in $\mathcal{K}$. Therefore, if for many different $\boldsymbol{k'}$ such that $T(\boldsymbol{k'}) = 0$, $T(\boldsymbol{k}+\boldsymbol{k'}) = 0$, then $\boldsymbol{k}$ is in $\mathcal{K}$ with high probability.

We make the hypothesis that for any fixed value $\boldsymbol{k}$ and random value $\boldsymbol{k'}$ the probability that $T(\boldsymbol{k}+\boldsymbol{k'}) = 0$ is independent of the probability that $T(\boldsymbol{k'}) = 0$. Under this hypothesis, we compute $p(\boldsymbol{k}) = \Pr[T(\boldsymbol{k} + \boldsymbol{k'}) = 0 \,/\, T(\boldsymbol{k'}) = 0]$.

For a random $\boldsymbol{k}$, the value $\boldsymbol{k}+\boldsymbol{k'}$ when $T(\boldsymbol{k'}) = 0$ is uniformly distributed and $p(\boldsymbol{k}) = \alpha$. However, if $\boldsymbol{k} \in \mathcal{K}$, then one can write $p(\boldsymbol{k}) = \Pr[\boldsymbol{k'} \in \mathcal{K} \,/\, T(\boldsymbol{k'}) = 0] + \Pr[\boldsymbol{k'} \notin \mathcal{K} \,/\, T(\boldsymbol{k'}) = 0]. \Pr[T(\boldsymbol{k} + \boldsymbol{k'}) = 0 \,/\, \boldsymbol{k} + \boldsymbol{k'} \notin \mathcal{K}] = \frac{\beta}{\alpha} + \frac{\alpha-\beta}{\alpha}\frac{\alpha-\beta}{1-\beta}$.

Under the hypothesis that $\beta \ll \alpha$, if $\boldsymbol{k} \in \mathcal{K}$ then $p(\boldsymbol{k})/\alpha = \frac{(1-\beta/\alpha)^2}{1-\beta} + \frac{\beta}{\alpha^2} \simeq 1 + \beta(\alpha^{-1} - 1)^2$. Therefore the difference between the values of $p(\boldsymbol{k})$ depending on whether $\boldsymbol{k} \in \mathcal{K}$ or not is of the order of $\alpha\beta$ and, by taking $N = 1/(\alpha\beta)^2$ elements $\boldsymbol{k'}$ such that $T(\boldsymbol{k'}) = 0$ and computing the average of $T(\boldsymbol{k}+\boldsymbol{k'})$, we can decide whether $\boldsymbol{k} \in \mathcal{K}$ or not. The complexity of this test is about β^{-2}.

We checked experimentally this hypothesis, for the parameters $\ell = 41$, $n = 137$ and $r = 6$. Testing if $p(\boldsymbol{k})/\alpha - 1 > \frac{1}{2}\beta(\alpha^{-1} - 1)^2$ is not sufficient to have an error-free test of membership of $\mathcal{K}$. However, testing if $p(\boldsymbol{k})/\alpha - 1 > \beta(\alpha^{-1} - 1)^2$ appear to be sufficient to detect about half of the members of $\mathcal{K}$.

Each value $\boldsymbol{k}$ has a probability q^{-r} of being in $\mathcal{K}$ and we need n distinct elements of $\mathcal{K}$. The whole complexity for finding $\mathcal{K}$ is nq^{3r}.

Technique 2.

In this technique, we define a graph whose vertices are the elements $\boldsymbol{k}$ such that $T(\boldsymbol{k}) = 0$, *i.e.* elements that may be in the kernel. For each pair $(\boldsymbol{k}, \boldsymbol{k'})$ of vertices, we compute $T(\boldsymbol{k} + \boldsymbol{k'})$. If the result is 0, then we put an edge between these two vertices. All vertices such that $\boldsymbol{k} \in \mathcal{K}$ are connected, *i.e.* the elements of $\mathcal{K}$ are in a large clique In practice, we don't construct the whole graph. We construct its restriction to N vertices. We are looking for vertices that correspond to $n - r$ independent elements of $\mathcal{K}$. If $N > n/\beta$, it is likely that the graph contains such vertices. The clique containing the elements of $\mathcal{K}$ contains at least βN vertices. Under the same hypothesis as above, that the probability that $T(\boldsymbol{k} + \boldsymbol{k'}) = 0$ is independent of the probability that $T(\boldsymbol{k}) = 0$, this graph restricted to N vertices has αN^2 edges. Apart from the vertices that correspond to elements of $\mathcal{K}$, the edges are randomly distributed. General results on random graph [44] gives us that the expected number of vertex in the clique of maximal order in random graph of N vertex with a probability α between each edge is $\frac{2 \log N}{\log 1/\alpha} + O(\log \log N)$. Therefore, if βN is significantly greater than $\frac{2 \log N}{\log 1/\alpha}$, then there will be a unique large clique, that gives a basis of $\mathcal{K}$. When $\beta \ll \alpha$, this condition is equivalent to $N \approx \beta^{-1} \log \beta^{-1}$ and the whole complexity for finding $\mathcal{K}$ is $q^{2r} \log^2 q^r$.

However, although this technique seems to be better than the previous one, we do not know a max-clique algorithm that benefits from the fact that we have a random and dense graph which has a very large clique. In practice, as we said before, a concrete attack of the PMI cryptosystem will use a mix of technique 1 (to find some elements very likely to be members of $\mathcal{K}$) and technique 2 (to extract from them a large clique).

A.4.4 Recovering the plaintext

Assume we have correctly found the kernel $\mathcal{K}$. Now, we have to reconstruct a family of n bilinear equations in the $\boldsymbol{x}$ and $\boldsymbol{y}$ variable for each affine subspace parallel to $\mathcal{K}$. When this has been done, then for fixed $\boldsymbol{y}$ we can try to solve each system in the $\boldsymbol{x}$ unknowns and decide the correct solution using redundancy.

The question one may ask is whether we still find $n - \gcd(\ell, n)$ independent equations for each affine subspace. What can be said is that the original n equations from the MI scheme are clearly friend when $\boldsymbol{x}$ is restricted to a subspace. Accordingly the number of independent equations can only increase, which is in favour of the attacker. Now, given a ciphertext $\boldsymbol{y}$, its corresponding plaintext is in some subspace parallel to $\mathcal{K}$ and for such ciphertext, each family of equations allow to recover at least $n - \gcd(\ell, n)$ coordinates of $\boldsymbol{x}$. Finally, an exhaustive search allows us to find the missing coordinates in time $q^{\gcd(\ell,n)}$ as well as the correct subspace to choose.

A.5 Alternative attack against the MI scheme

In this section, we show a new attack against the MI scheme. We apply the same technique as in the PMI scheme. First of all, we compute the differential and next we study the kernel of the transpose of this application. In order to simplify the exposition of the attack, we assume in the following that $\gcd(\ell, n) = 1$ and $q = 2$.

A.5.1 Overview

As for Patarin's attack, this attack tries to find n bilinear forms in the ciphertext coordinates and in a vector related to the plaintext. Next, when a ciphertext is given, the

n linear equations in the vector related to the plaintext allow us to recover this vector. Finally, since this vector is related to the plaintext by a linear system, we can easily decrypt.

More precisely, the attacks computes two bilinear systems, $C(\boldsymbol{x}, \boldsymbol{y})$ and $D(\boldsymbol{x}, \boldsymbol{y})$, such that for ${}^t\boldsymbol{f_k}$ in the kernel of ${}^t\boldsymbol{L_{E,k}}$ we have

$$C(\boldsymbol{E}(\boldsymbol{k}), \boldsymbol{f_k}) = 0 \text{ and } D(\boldsymbol{k}, \boldsymbol{f_k}) = 0$$

This allows to compute $\boldsymbol{k}$ from $\boldsymbol{E}(\boldsymbol{k})$.

A.5.2 Description

For the MI scheme, the differential can be written as

$$L_{F,k}(x) = x^{q^\ell} \cdot k + x \cdot k^{q^\ell} = k^{q^\ell+1} \cdot \left(\frac{x}{k} + \left(\frac{x}{k} \right)^{q^\ell} \right)$$

If we define the following three *linear* functions over $\mathbb{F}_{q^n}$: $\mu_k(x) = F(k) \cdot x$, where $F(k) = k^{q^\ell+1}$, $\psi(x) = x^{q^\ell} + x$ and $\theta_k(x) = \frac{x}{k}$, then

$$L_{F,k} = \mu_k \circ \psi \circ \theta_k$$

Let us define $\boldsymbol{\mu_k} = \pi \circ \mu_k \circ \pi^{-1}$, $\boldsymbol{\psi} = \pi \circ \psi \circ \pi^{-1}$ and $\boldsymbol{\theta_k} = \pi \circ \theta_k \circ \pi^{-1}$ for $k = \pi^{-1}(\boldsymbol{S}(\boldsymbol{k}))$. Therefore

$$\boldsymbol{L_{E,k}} = \boldsymbol{T} \circ \boldsymbol{\mu_k} \circ \boldsymbol{\psi} \circ \boldsymbol{\theta_k} \circ \boldsymbol{S}$$

where all terms are linear functions of $\mathbb{F}_q^n$. The matrix of $\boldsymbol{L_{F,k}}$ is a product of $n \times n$ matrices of $\mathbb{F}_q$.

Let ${}^t\boldsymbol{f_k}$ be in the kernel of the transpose ${}^t\boldsymbol{L_{E,k}}$. This means that the product $(\boldsymbol{f_k})(\boldsymbol{L_{E,k}})$ is the null vector $\mathbf{0}$, which is equivalent to

$$(\boldsymbol{f_k})(\boldsymbol{T}.\boldsymbol{\mu_k}.\boldsymbol{\psi}.\boldsymbol{\theta_k}.\boldsymbol{S}) = \mathbf{0}$$

where $\boldsymbol{f_k}$ is a n-dimensional row vector and $\boldsymbol{T}$, $\boldsymbol{\mu_k}$, $\boldsymbol{\theta_k}$, and $\boldsymbol{S}$ are $n \times n$ invertible matrices and $\boldsymbol{\psi}$ is a $n \times n$ matrix. Since $\boldsymbol{\theta_k}$ and $\boldsymbol{S}$ are one-to-one, this is equivalent to $(\boldsymbol{f_k})(\boldsymbol{T}.\boldsymbol{\mu_k}) \in \mathrm{Ker}\, {}^t\boldsymbol{\psi}$, the application ${}^t\boldsymbol{\psi}$ being independent of $\boldsymbol{k}$.

Recall that in the case where $\gcd(\ell, n) = 1$ the kernel of $\boldsymbol{L_{E,k}}$ is of dimension 1 and is generated by $\boldsymbol{k}$. The transpose ${}^t\boldsymbol{L_{E,k}}$ also has a kernel of dimension 1. The kernel of ${}^t\boldsymbol{\psi}$ is one-dimensional and independent of $\boldsymbol{k}$. Therefore if $q = 2$, $\mathrm{Ker}\, {}^t\boldsymbol{\psi} = \{\mathbf{0}, \hat{\boldsymbol{f}}\}$ and the previous equation can be rewritten as $(\boldsymbol{f_k})(\boldsymbol{T}.\boldsymbol{\mu_k}) = (\hat{\boldsymbol{f}})$.

From $\mu_k(x) = F(k) \cdot x$, we deduce that $\boldsymbol{\mu_k}$ is linear in

$$F(k) = F(\pi^{-1}(\boldsymbol{S}(\boldsymbol{k}))) = \pi^{-1}(\boldsymbol{T}^{-1}(\boldsymbol{E}(\boldsymbol{k})))$$

i.e. linear in $\boldsymbol{E}(\boldsymbol{k})$, and therefore the equation $(\boldsymbol{f_k})(\boldsymbol{T}.\boldsymbol{\mu_k}) = (\hat{\boldsymbol{f}})$ is bilinear in $\boldsymbol{f_k}$ and $\boldsymbol{E}(\boldsymbol{k})$. Accordingly whenever a ciphertext $\boldsymbol{y} = \boldsymbol{E}(\boldsymbol{k})$ is given, the corresponding $\boldsymbol{f_k}$ can be found by solving a linear system.

Finally, as $(\boldsymbol{f_k})(\boldsymbol{L_{E,k}}) = 0$ and $\boldsymbol{L_{E,k}}$ is linear in the $\boldsymbol{k}$ variable we have again a bilinear relation between $\boldsymbol{k}$ and $\boldsymbol{f_k}$. Now, since $\boldsymbol{f_k}$ is known from $\boldsymbol{y}$, we get a system with n equations in n coordinates of the variable $\boldsymbol{k}$. This system has a kernel of dimension one, and consequently, we can easily decrypt.

Acknowledgement

This work is supported in part by the French government through X-Crypt and in part by the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT.

A.6 Appendix : Some useful mathematical results

Lemme A.6. *For any integers q, i and n,* $\gcd(q^n-1, q^i-1) = q^{\gcd(n,i)}-1$ ▽

Démonstration. Let $(r_k)_{k\geq 0}$ be the sequence of integers obtained by the Euclidean algorithm from $r_0 = n$ and $r_1 = i$. If k_0 is the largest integer such that $r_{k_0} \neq 0$, then $r_{k_0} = \gcd(n, i)$.

Similarly, let $(R_k)_{k\geq 0}$ be the sequence of polynomials obtained from the Euclidean algorithm from $R_0 = X^n - 1$ and $R_1 = X^i - 1$. We recall that n_1 is the largest integer such that $R_{n_1} = \gcd(X^n-1, X^i-1)$. We show by recurrence on n that for $0 \leq k \leq k_0+1$, $R_k = X^{r_k} - 1$. It is correct by assumption for $k = 0$ and $k = 1$. Assuming that $k \geq 2$ and $k \leq k_0 + 1$. Let us write $r_{k-2} = \alpha r_{k-1} + r_k$. Then,

$$\begin{aligned} X^{r_{k-2}} - 1 &= (X^{r_{k-1}} - 1)(X^{r_{k-2}-r_{k-1}} + X^{r_{k-2}-2r_{k-1}} + \cdots + X^{r_{k-2}-\alpha r_{k-1}}) \\ &\quad + X^{r_k} - 1 \end{aligned}$$

Therefore, $X^{r_k}-1$ is the remainder of the division of $R_{k-2} = X^{r_{k-2}}-1$ by $R_{k-1} = X^{r_{k-1}}-1$ since $r_k < r_{k-1}$. So, $R_{k_0+1} = X^0 - 1 = 0$ and $R_{k_0} \neq 0$. Consequently, $k_1 = k_0$ and $R_{k_1} = R_{k_0} = X^{r_{k_0}} - 1 = X^{\gcd(i,n)} - 1$. If we replace X by q, we get the lemma. ∎

The following lemma is useful to exactly estimate the kernel dimension. We require exact value and not upper bounds on the number of solutions as done in [238].

Lemme A.7. *In a finite field $\mathbb{F}_{q^n}$ with q^n elements, the equation $X^j = A$ has either* 0 *solution or* $\gcd(j, q^n-1)$ *solutions.* ▽

Démonstration. The multiplicative group of the finite field $\mathbb{F}_{q^n}$ has $q^n - 1$ elements. The simple case is when $\gcd(j, q^n - 1) = 1$. Therefore, j is invertible modulo $(q^n - 1)$ and we denote by h the inverse of j. Then, if we raise the equation $X^j = A$ to the power h, we get $X = X^{jh} = A^h = A'$, and so there is only one solution.

On the other hand, if $\gcd(j, q^n - 1) = d \neq 1$. Let $j' = j/d$, then $\gcd(j', q^n - 1) = 1$ and let h' be the inverse of j' modulo $q^n - 1$. We can raise the equation to the power h' and get $X^d = X^{jh'} = X^{j'dh'} = A^{h'} = A'$. This equation may have no solution if A' is not a d-th power of some value of $\mathbb{F}_{q^n}$. We now show that the equation $X^d = A'$ has d solutions when A' is a d-th power. We know that there is at least one solution which can be found by a randomised algorithm of Adleman, Manders and Miller. The other solutions are obtained by multiplying the original solution by the d roots of unity. We finally explain why there are d d-th roots of unity. Since the multiplicative group of a finite field is a cyclic group, there is a primitive element g, that generates the whole group. Therefore, $g' = g^{\frac{q^n-1}{d}}$ is a d-th root of unity and for $0 \leq i < d$, g'^i ranges over the set of all roots. This completes the proof of the lemma. ∎

If $j = q^i - 1$, then we can combine both lemmas. In a finite field $\mathbb{F}_{q^n}$ with q^n elements, the equation $X^{q^i-1} = A$ has either 0 solution or $q^{\gcd(i,n)} - 1$ solutions.

Annexe B

Practical Cryptanalysis of SFLASH

CRYPTO 2007
[96] avec Vivien Dubois, Adi Shamir et Jacques Stern

Abstract : *In this paper, we present a practical attack on the signature scheme SFLASH proposed by Patarin, Goubin and Courtois in 2001 following a design they had introduced in 1998. The attack only needs the public key and requires about one second to forge a signature for any message, after a one-time computation of several minutes. It can be applied to both SFLASH^{v2} which was accepted by NESSIE, as well as to SFLASH^{v3} which is a higher security version.*

B.1 Introduction

In the last twenty years, multivariate cryptography has emerged as a potential alternative to RSA or DLOG [257, 133] schemes. Many schemes have been proposed whose security appears somehow related to the problem of deciding whether or not a quadratic system of equations is solvable, which is known to be NP-complete [134]. An attractive feature of such schemes is that they have efficient implementations on smart cards, although the public and secret keys are rather large. Contrary to RSA or DLOG schemes, no polynomial quantum algorithm is known to solve this problem.

The SFLASH Scheme.

SFLASH is based on the Matsumoto-Imai scheme (MI) [212], also called the C^* scheme. It uses exponentiation $x \mapsto x^{q^\theta+1}$ in a finite field $\mathbb{F}_{q^n}$ of dimension n over a binary field $\mathbb{F}_q$, and two affine maps on the input and output variables. The MI scheme was broken by Patarin in 1995 [238]. However, based on an idea of Shamir [268], Patarin *et al.* proposed at CT-RSA 2001 [241] to remove some equations from the MI public key and called the resulting scheme C^{*-}. This completely avoids the previous attack and, although not appropriate for an encryption scheme, it is well-suited for a signature scheme. The scheme was selected in 2003 by the NESSIE European Consortium as one of the three recommended public key signature schemes, and as the best known solution for low cost smart cards.

Previous Attacks on SFLASH.

The first version of SFLASH, called SFLASH^{v1}, is a more efficient variant of C^{*-} using a small subfield. It has been attacked by Gilbert and Minier in [137]. However, the later versions (SFLASH^{v2} and SFLASH^{v3}) were immune to this attack.

Recently, Dubois, Fouque and Stern in [97] proposed an attack on a special class of SFLASH-like signatures. They show that when the kernel of the linear map $x \mapsto x + x^{q^\theta}$ is non-trivial, the C^{*-} scheme is not secure. The attack is very efficient in this case, but relies on some specific properties which are not met by the NESSIE proposals and which make the scheme look less secure.

Our Results.

In this paper, we achieve a total break of the NESSIE standard with the actual parameters suggested by the designers : given only the public key, a signature for any message can be forged in about one second after a one time computation of several minutes. The asymptotic running time of the attack is $O(\log^2(q)n^6)$ since it only needs standard linear algebra algorithms on $O(n^2)$ variables, and n is typically very small. As in [97], the basic strategy of the attack is to recover additional independent equations in order to apply Patarin's attack [238]. To this end, both attacks use the differential of the public key. However, the attacks differ in the way the invariants related to the differential are found. The differential of the public key, also called its polar form, is very important since it transforms quadratic equations into linear ones. Hence, it can be used to find some linear relations that involve the secret keys. Its cryptanalytic significance had been demonstrated in [110].

Organization of the Paper.

In section 2, we describe the SFLASH signature scheme and the practical parameters recommended by Patarin *et al.* and approved by NESSIE. Then, in section 3 we present the multiplicative property of the differential that we need. Next, in section 4 we describe how to recover linear maps related to multiplications in the finite field from the public key. In section 5, we show how to break the NESSIE proposal given only the public key. In section 6, we extend the attack to cover the case when up to half of the equations are removed, and finally in section 7, we compare our method with the technique of [97] before we conclude.

B.2 Description of SFLASH

In 1988, Matsumoto and Imai [212] proposed the C^* scheme for encryption and signature. The basic idea is to hide a quadratic easily invertible mapping F in some large finite field $\mathbb{F}_{q^n}$ by two secret invertible linear (or affine) maps U and T which mix together the n coordinates of F over the small field $\mathbb{F}_q$:

$$P = T \circ F \circ U$$

where $F(x) = x^{q^\theta+1}$ in $\mathbb{F}_{q^n}$. This particular form was chosen since its representation as a multivariate mapping over the small field is quadratic, and thus the size of the public key is relatively small.

The secret key consists of the maps U and T ; the public key P is formed by the n quadratic expressions, whose inputs and outputs are mixed by U and T, respectively. It

can be seen that F and P are invertible whenever $\gcd(q^\theta + 1, q^n - 1) = 1$, which implies that q has to be a power of 2 since q is a prime power.

This scheme was successfully attacked by Patarin [238] in 1996. To avoid this attack and restore security Patarin *et al.* proposed in [242] to remove from the public key the last r quadratic expressions (out of the initial n), and called this variant of C^* schemes, C^{*-}. Furthermore, if the value of r is chosen such that $q^r \geq 2^{80}$, then the variant is termed C^{*--}. If we denote by Π the projection of n variables over $\mathbb{F}_q$ onto the first $n - r$ coordinates, we can represent the public key by the composition :

$$P_\Pi = \Pi \circ T \circ F \circ U = T_\Pi \circ F \circ U.$$

In the sequel, P denotes the public key of a C^* scheme whereas P_Π denotes a C^{*-} or C^{*--} public key. In both cases the secret key consists of the two linear maps T and U.

To sign a message m, the last r coordinates are chosen at random, and the signer recovers s such that $P_\Pi(s) = m$ by inverting of T, U and F. A signature (m, s) can be checked by computing $P_\Pi(s)$ with the public key, which is extremely fast since it only involves the evaluation of a small number of quadratic expressions over the small finite field $\mathbb{F}_q$.

For the NESSIE project and in [241], Patarin *et al.* proposed two particular recommended choices for the parameters of C^{*--} :

– for SFLASH^{v2} : $q = 2^7$, $n = 37$, $\theta = 11$ and $r = 11$
– for SFLASH^{v3} : $q = 2^7$, $n = 67$, $\theta = 33$ and $r = 11$

SFLASH^{v3} was actually proposed to provide an even more conservative level of security than SFLASH^{v2} [241]. However, the designers made clear that they viewed SFLASH^{v2} as providing adequate security, and no attack on these two choices of parameters had been reported so far.

The important fact to notice here is that in both cases $\gcd(n, \theta) = 1$ and thus the attack described in [97] on a modified version of SFLASH in which $\gcd(n, \theta) > 1$ cannot be applied. The attack described in this paper shares with [97] the basic observation about the multiplicative property of C^{*-} schemes which is described in section 3, but proceeds in a completely different way. More discussion about the relationships between the two attacks can be found in section 7.

B.3 The Multiplicative Property of the Differential

The attack uses a specific multiplicative property of the differential of the public key of a C^{*-} scheme.

The differential of the internal quadratic system $F(x) = x^{q^\theta+1}$ is a symmetric bilinear function in $\mathbb{F}_{q^n}$, called DF, and it is defined for all $a, x \in \mathbb{F}_{q^n}$ by the linear operator :

$$DF(a, x) = F(a + x) - F(a) - F(x) + F(0).$$

When $F(x) = x^{q^\theta+1}$, we get for all $a, x \in \mathbb{F}_{q^n}$

$$DF(a, x) = ax^{q^\theta} + a^{q^\theta}x.$$

Note that this expression is bilinear since exponentiation by q^θ is a linear operation. This map has a very specific multiplicative property : for all $\xi \in \mathbb{F}_{q^n}$

$$DF(\xi \cdot a, x) + DF(a, \xi \cdot x) = (\xi + \xi^{q^\theta}) \cdot DF(a, x) \tag{B.1}$$

We now explain how this identity on the internal polynomial induces a similar one on the differential of the public keys in C^* and C^{*-}. Due to the linearity of the DP operator, we can combine it with the linear maps T and U to get that the differential of any C^* public key P is $DP(a,x) = T \circ DF(U(a), U(x))$. Then, equation (B.1) becomes for any $\xi \in \mathbb{F}_{q^n}$:

$$\begin{aligned} T \circ DF(\xi \cdot U(a), U(x)) &+ T \circ DF(U(a), \xi \cdot U(x)) \\ &= T \circ (\xi + \xi^{q^\theta}) \cdot DF(U(a), U(x)) \\ &= T \circ (\xi + \xi^{q^\theta}) \cdot T^{-1}(DP(a,x)). \end{aligned}$$

We denote by M_ξ and $M_{L(\xi)}$ respectively the multiplications by ξ and by $L(\xi) = \xi + \xi^{q^\theta}$. Also, we let N_ξ denote the linear map $U^{-1} \circ M_\xi \circ U$ which depends on the secret key. We still use the word "multiplication" for N_ξ, even though this wording is not actually accurate since this is not the standard multiplication in $\mathbb{F}_{q^n}$, due to the action of the input transformation U. With these notations :

$$DP(N_\xi(a), x) \quad + \quad DP(a, N_\xi(x)) = T \circ M_{L(\xi)} \circ T^{-1}(DP(a,x)).$$

Finally, if DP_Π is the differential of a C^{*-} public key P_Π, then :

$$DP_\Pi(N_\xi(a), x) + DP_\Pi(a, N_\xi(x)) = T_\Pi \circ M_{L(\xi)} \circ T^{-1}(DP(a,x)).$$

Let $\Lambda(L(\xi))$ denote the linear map $T_\Pi \circ M_{L(\xi)} \circ T^{-1}$, then

$$DP_\Pi(N_\xi(a), x) + DP_\Pi(a, N_\xi(x)) = \Lambda(L(\xi))\left(DP(a,x)\right). \tag{B.2}$$

This last equation is interesting since each coordinate of the left hand side is linear in the unknown coefficients of N_ξ and each coordinate of the right hand side is a linear combination by the unknown coefficients of $\Lambda(L(\xi))$ of the symmetric bilinear coordinate forms of the original DP, which are partially known since their first $(n-r)$ coordinates are public.

The heart of the attack consists in identifying some N_ξ, given the public key and equation (B.2), and then using its mixing effect on the n coordinates to recover the r missing quadratic forms from the $(n-r)$ known quadratic forms of the public key. In the next section, we will see how to recover some non-trivial multiplication N_ξ, in which ξ can be any value in $\mathbb{F}_{q^n} \setminus \mathbb{F}_q$.

B.4 Recovering Multiplications from the Public Key

Any linear mapping can be represented by an $n \times n$ matrix with n^2 entries from $\mathbb{F}_q$. Note that the multiplications N_ξ form a tiny subspace of dimension n within the space of all linear maps whose dimension is n^2.

The coordinates of DP_Π are known symmetric bilinear forms that can be seen as $n(n-1)/2$-dimensional vectors. They generate a $(n-r)$-dimensional subspace V_Π which is contained in the n-dimensional space V, generated by the full set of coordinates of DP in the original C^* public key.

Consider now the expression :

$$S_M(a,x) = DP_\Pi(M(a), x) + DP_\Pi(a, M(x))$$

where S_M is defined for any linear mapping M as a $(n-r)$-tuple of symmetric bilinear forms. Most choices of M do not correspond to any multiplication by a large field element

ξ, and thus we do not expect them to satisfy the multiplicative property described in section 3. Due to relation (B.2), when M is a multiplication N_ξ, the $(n-r)$ coordinates of S_{N_ξ} are in V. It is unlikely that they are all in the subspace V_Π. However, there is a huge number of possible values for ξ, and it can be expected that for some choices of $\xi \in \mathbb{F}_{q^n} \setminus \mathbb{F}_q$, some of the bilinear forms in $S_M(a,x)$ will be contained in the known subspace V_Π. Our goal now is to detect such special multiplications.

Dimension of the overall linear maps space.

Let us consider k of the published expressions, for instance the first k, and let us study the vector space $E(1,\dots,k)$ of linear maps M such that the first k coordinates of $S_M(a,x)$ are all contained in V_Π. Since membership in V_Π is expressed by the vanishing of $n(n-1)/2-(n-r)$ linear forms, the elements of this subspace satisfy a system of $k\cdot(n(n-1)/2-(n-r))$ linear equations in the n^2 unknown coefficients of M. If all these equations were independent, the dimension of $E(1,\dots,k)$ would be $n^2 - k\cdot(n(n-1)/2-(n-r))$ which is clearly impossible as soon as $k \geq 3$. Otherwise, we can only claim that it is lower-bounded by this number. On the other hand, it can be seen that the space $E(1,\dots,k)$ contains a subspace of multiplications, whose dimension is now to be computed.

Dimension of the multiplications space.

For a multiplication N_ξ, thanks to equation (B.2), the coordinates of S_{N_ξ} are guaranteed to be linear combinations of the coordinates of DP, whose coefficients $\Lambda(L(\xi))$ are linear in $\xi + \xi^{q^\theta}$. Setting $\zeta = \xi + \xi^{q^\theta}$, the first k linear combinations are given by the k linear forms

$$\Lambda_i(\zeta) = \Pi_i \circ T \circ M_\zeta \circ T^{-1}$$

for $i = 1,\dots,k$ where Π_i is the projection on the ith coordinate. Note that $\Lambda_i : \zeta \mapsto \Lambda_i(\zeta)$ are linear bijections from $\mathbb{F}_{q^n}$ to $(\mathbb{F}_q^n)^*$, the vector space of linear forms over $\mathbb{F}_q^n$. Indeed, the kernel of Λ_i consists of the elements ζ such that the ith row of $T \circ M_\zeta \circ T^{-1}$ is zero. Since $T \circ M_\zeta \circ T^{-1}$ is invertible for $\zeta \neq 0$, the kernel of Λ_i must be trivial. This implies that Λ_i is a linear bijection, and we will use this property. Note that this is the converse of the assumption underlying the attack in [97], and in this sense, our new attack and the old attack can be seen as complementary.

Let us consider the subspace L' of $(\mathbb{F}_q^n)^*$ generated by the first $(n-r)$ coordinate projections. In this case, the k conditions $\Pi_i \circ S_{N_\xi} \in V_\Pi$ become

$$\Lambda_i(L(\xi)) \in L',\ \forall i = 1,\dots,k \tag{B.3}$$

which means that $\Lambda_i(L(\xi))$ only depends on the $(n-r)$ first rows of DP, *i.e.* only on the known DP_Π.

Consequently, when searching for a multiplication by ξ for which equation (B.3) holds, we get the following set of conditions on $\zeta = L(\xi) = \xi + \xi^{q^\theta}$:

$[(i)]$

$\zeta \in \mathrm{Im}(L)$

2. $\Lambda_i(\zeta) \in L'$ for $i = 1,\dots,k$

Since $\zeta = \xi + \xi^{q^\theta}$ and $\gcd(n,\theta) = 1$, ζ is non-zero unless $\xi = 0$ or 1. This means that the kernel of L has dimension 1, hence ζ ranges over a space of dimension $n-1$. Condition (i) corresponds to a single linear relation over the coordinates of $L(\xi)$, since

$\dim \mathrm{Im}(L) = n - 1$. Also, since Λ_i is a linear bijection and L' is of codimension r, each of the conditions in (ii) corresponds to r additional linear relations. Altogether, this means that we have $kr + 1$ linear equations. Furthermore, since we are interested in the space of N_ξ's and not in the space of M_ζ's, the dimension is $n - kr - 1 + 1 = n - kr$ since the kernel of L is of dimension 1. This implies that whenever we add a condition (*i.e.* increase k by 1), we add about $n^2/2$ linear equations on the full space of linear maps, but their effect on the subspace of multiplications is to reduce its dimension only by r. Finally, the space of multiplications in $E(1, \ldots, k)$ includes at least one non-trivial multiplication, *i.e.* a multiplication by an element outside $\mathbb{F}_q$ whenever

$$n \geq kr + 2. \tag{B.4}$$

Consequently, the dimension of $E(1, \ldots, k)$ is

$$\max \left\{ n^2 - k \left(\frac{n(n-1)}{2} - (n-r) \right), n - kr, 1 \right\}.$$

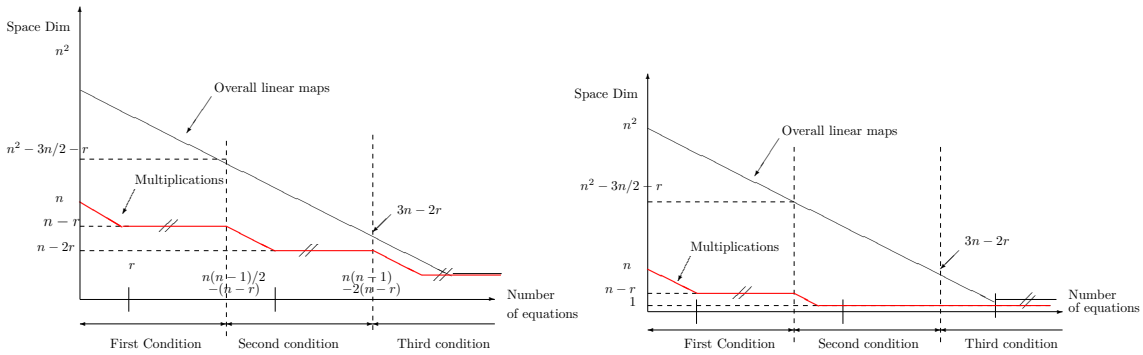

FIGURE B.1 – Evolution of the dimensions of the overall linear maps and their subspace of multiplications when $r < n/3$ and when $r \geq n/3$, as we add more linear equations.

Figure B.1 describes the expected evolution of the dimension of the space of all linear maps and of the dimension of the subspace of multiplications for two different choices of r. The intuition behind our attack is that initially there are many "useless maps" and few multiplications. However, the number of useless maps drops rapidly as we add more equations, whereas the number of multiplications drops slowly (since many of the equations are linearly related on the subspace of multiplications). This leads to an elimination race, and we hope to get rid of all the "bad maps" before we inadvertantly kill off all the "good maps" by imposing too many conditions.

Taking $k = 3$, it can be seen that the first expression of the max is not positive. This seems to indicate that $E(1, \ldots, k)$ consists entirely of multiplications. This is demonstrated in the left figure. This subspace contains non-trivial multiplications, whenever $n - 3r > 1$. Therefore, the attack is expected to work for values of r up to $(n-2)/3$. The right figure shows a case in which r is too large, and thus the "good maps" are eliminated before the "bad maps". We will see in section B.6 how to improve the attack and deal with values of r up to about $n/2$. Note that even without this improvement, our technique is already sufficient to recover non-trivial multiplications for the recommended parameters of SFLASH^{v2} and SFLASH^{v3}, since $r = 11$ is smaller than both $35/3$ and $65/3$. Of course, the argument that was offered is only heuristic. However, it was confirmed by a large number of experiments, in which the attack always behaved as expected by our heuristic analysis, and signatures were successfully forged.

B.5 Recovering a Full C^* Public Key

The final part of the attack is to recover a set P'_Π of additional equations which are independent of the first system P_Π. If the rank of the concatenation of the original P_Π and the newly computed r equations of P'_Π is full, then Patarin's attack on MI [238] can be mounted, although we do not necessarily reconstruct the r original equations of the full public key. This idea is the same as in [97].

Recovering a full rank system.

To reconstruct a full rank system, we note that the action of the final linear map T is to compute different linear combinations of the full (*i.e.* non-truncated) internal quadratic polynomials $F \circ U$. Consequently, if we were able to mix by some linear mapping the internal quadratic coordinates $F \circ U$ before the action of T_Π, then we will be able to create new quadratic polynomials which could replace the r missing ones.

When we compose the multiplication $N_\xi = U^{-1} \circ M_\xi \circ U$ (which was found in the previous part of the attack) with the truncated public key P_Π, the inputs of the internal quadratic mapping $F(x) = x^{q^\theta+1}$ are multiplied by ξ. Indeed,

$$P_\Pi \circ N_\xi = T_\Pi \circ F \circ M_\xi \circ U$$

since $P_\Pi \circ N_\xi(x) = T_\Pi \circ F \circ U \circ U^{-1} \circ M_\xi(U(x)) = T_\Pi(F(M_\xi(U(x))))$. Let us denote this new system by P'_Π. We can show that the outputs of the internal quadratic equations $F \circ U$ are multiplied by $\xi^{q^\theta+1}$. Indeed, $T_\Pi \circ F(\xi \cdot U(x)) = T_\Pi((\xi \cdot U(x))^{q^\theta+1}) = T_\Pi(\xi^{q^\theta+1} \cdot F(U(x)))$, and so :

$$P'_\Pi = P_\Pi \circ N_\xi = T_\Pi \circ M_{\xi^{q^\theta+1}} \circ F \circ U$$

Let us consider the special case $\xi \in \mathbb{F}_{q^n} \setminus \mathbb{F}_q$. In this situation, we say that N_ξ is non-trivial. Since F is a permutation and thus $F(\mathbb{F}_q) = \mathbb{F}_q$, $\xi^{q^\theta+1}$ is not in $\mathbb{F}_q$ either. Thus, the multiplication by $M_{\xi^{q^\theta}+1}$ is non-trivial, *i.e.* corresponds in particular to a non-diagonal matrix.

Therefore, in the sets P_Π and P'_Π the internal quadratic coordinates of $F \circ U$ are mixed with two different linear combinations, T_Π and $T_\Pi \circ M_{\xi^{q^\theta}+1}$. We hope that for some value $\xi \in \mathbb{F}_{q^n} \setminus \mathbb{F}_q$, r equations in the set P'_Π together with P_Π will form a full rank system. This special case is not necessary since we could use different values of ξ to add r different quadratic forms to the $(n-r)$ public ones. However, in our experiments it was always sufficient to use one ξ, and then Patarin's attack could be applied to forge actual signatures.

In practice, to determine if the new system of n equations is of full rank, we simply tested whether Patarin's attack succeeded. If not, another set of r equations was chosen amongst the $(n-r)$ equations of P'_Π. For each choice of r equations, the success probability was approximately $1 - 1/q$, which is close to 1 for $q = 2^7$

If $\xi \in \mathbb{F}_q$ (*i.e.* the multiplication is trivial), P'_Π is simply P_Π where each coordinate has been multiplied by the same element of $\mathbb{F}_q$, since $F(\mathbb{F}_q) = \mathbb{F}_q$ and multiplication by an element of $\mathbb{F}_q$ is a diagonal matrix. Thus, such trivial ξ are not interesting for our attack and this is the reason why they were discarded from our search for appropriate N_ξ in the previous section.

Practical results.

We carried our experiments on a 2GHz AMD Opteron PC using different parameters. The following table provides the time to recover a non-trivial multiplication and the time

to recover an independent set of equations which form a full rank system. This computation has to be done only once per public key. Then Patarin's attack requires about one second to forge an actual signature for any given message. All these operations can be carried out by solving various systems of linear equations with a relatively small number of variables ($O(n^2)$ or $O(n)$, depending on the operation).

The two columns in bold font represent the time to attack SFLASH^{v2} and SFLASH^{v3}. The notation 's' is for seconds and 'm' is for minutes.

n	37	**37**	67	**67**	131
θ	11	**11**	33	**33**	33
q	2	**128**	2	**128**	2
r	11	**11**	11	**11**	11
N_ξ Recovery	4s	**70s**	1m	**50m**	35m
C^* Recovery	7.5s	**22s**	2m	**10m**	7m
Forgery	0.01s	**0.5s**	0.02s	**2s**	0.1s

B.6 Breaking SFLASH when the Number of Deleted Quadratic Equations r is up to $n/2$

In this section, we deal with this problem by a technique which we call *distillation*, since it allows to gradually filter additional linear maps which are not multiplications. When $r \leq (n-2)/3$, we can use three conditions to eliminate all the useless linear maps, while retaining at least a two dimensional subspace of multiplications (since we reduce the initial n coordinates three times by r). When $r > (n-2)/3$, this will usually kill all the multiplications along with the useless linear maps.

Distillation is performed by relaxing the constraints, *i.e.* by forcing only two coordinates of S_M to be in V_Π. This will cancel a large fraction of useless linear maps, but not all of them. To clarify the situation, we use in the rest of this section angular brackets to demonstrate the stated number of dimensions for the SFLASH^{v3} parameters of $n = 67$ and $r = 11$.

After forcing the two conditions, the dimension of the space of linear maps is reduced to

$$n^2 - 2(n(n-1)/2 - (n-r)) = 3n - 2r \ \langle 179 \rangle$$

of the n^2 $\langle 4489 \rangle$ at the beginning, while the dimension of the good subspace (*i.e.* the subspace of multipications) is $n - 2r$ $\langle 45 \rangle$. Now, to find at least one non-trivial multiplication, we need to eliminate all the remaining useless linear maps. The new idea is that we can perform this process twice with different pairs of coordinates, *i.e.* coordinates 1 and 2 for the first time and coordinates 3 and 4 for the second, and get two different sets of linear maps, say VS_1 and VS_2, which contain both good and bad linear maps. Two random linear subspaces of dimension m in a linear space of dimension t are likely to have a nonzero intersection if and only if $m > t/2$, and then the dimension of the intersection is expected to be $2m - t$. We can apply this criterion separately to the space of all linear maps (in which $t = n^2$) and to the subspace of multiplications (in which $t = n$). In our example $VS_1 \cap VS_2$ is likely to contain non-trivial multiplications since $\langle 45 \rangle > \langle 67 \rangle / 2$, but is not likely to contain other maps since $\langle 179 \rangle < \langle 4489 \rangle / 2$. More generally, we may have to replace each one of SV_1 and SV_2 by the sum of several such linear subspaces in order to build up the dimension of the multiplications to more than $n/2$. For example, if each VS_i has only a $\langle 10 \rangle$-dimensional subspace of multiplications, we can replace it by the sum of four such linear subspaces to get the expected dimension up to $\langle 40 \rangle$, and the

intersection of two such sums will have an expected dimension of $\langle 13 \rangle$, and thus many non-trivial multiplications.

Asymptotic Analysis.

We now show how to deal with any $r < (1-\varepsilon)n/2$ for a fixed ε and large enough n. Note that our goal here is to simplify the description, rather than to provide the most efficient construction or tightest analysis. Since $n - 2r > \varepsilon n$, we can impose pairs of conditions and create linear subspaces VS_i of total dimension $O(n)$ which contain a subspace of multiplications of dimension $\varepsilon n \geq 2$. If we add $1/\varepsilon$ such subspaces, the dimension of the subspace of multiplications will increase to almost n, while the total dimension will remain n/ε, which is much smaller than n^2. Consequently, the intersection of two such sums is likely to consist entirely of multiplications.

Experimentations.

We get the following timing results when r is close to $n/2$ and 's', 'm' and 'h' respectively denotes seconds, minutes and hours.

n	37	37	67	67
θ	11	11	33	33
q	2	128	2	128
r	17	16	32	31
N_ξ Recovery	$8s$	$4m$	$3.5m$	$10h$
C^* Recovery	$7.5s$	$22s$	$3m$	$10m$
Forgery	$0.01s$	$0.4s$	$0.02s$	$2s$

B.7 Comparison with the Method of Dubois *et al.* [97]

In both attacks, the basic strategy is to recover additional independent equations in order to apply Patarin's attack [238]. They both use the differential of the public key, but differ in the way the invariants of the differential are found. The method of [97] can only deal with schemes where $\gcd(n, \theta) > 1$, which implies that the kernel of $L(\xi) = \xi + \xi^{q^\theta}$ is of dimension strictly larger than 1.

To recover non-trivial multiplication in [97], skew-symmetric mappings with respect to a bilinear form B are considered, *i.e.* linear maps M such that $B(M(a), x) = -B(a, M(x))$. In fact, the authors show that skew-symmetric mappings related to the symmetric bilinear forms of a C^* public key are specific multiplications in the extension $\mathbb{F}_{q^n}$ by means of a suitable transformation depending on the secret key, namely $U^{-1} \circ M_\xi \circ U$ where $\xi \in \operatorname{Ker} L$. For such maps, we get $DP(M(a), x) + DP(a, M(x)) = 0$. Since DP can be computed from the public key, this equation defines linear equations in the unknowns of M. However, in the case considered in this paper, *i.e.* when $\dim \operatorname{Ker} L = 1$ or equivalently when $\gcd(n, \theta) = 1$, the only skew-symmetric maps are the trivial multiplications which are useless to recover new independent quadratic equations.

To recover non-trivial multiplications, we introduce here different and more elaborate conditions related to the vector space generated by the various images of the differential in public key coordinates. In this case, we are also able to detect images of multiplications. However, the multiplications to be found are not known in advance but are only shown to exist by counting arguments, and the way we find them is by setting up an elimination race between the multiplications and other linear maps.

B.8 Conclusion

Multivariate cryptographic schemes are very efficient but have a lot of exploitable mathematical structure. Their security is not fully understood, and new attacks against them are found on a regular basis. It would thus be prudent not to use them in any security-critical applications.

One of the most interesting open problems is whether the new techniques described in this paper can be applied to the HFE cryptosystem [240]. The main attacks discovered so far against HFE are based on Gröbner Basis [102], and are very slow. So far, we could not find a way how to detect non-trivial multiplications in HFE, since it lacks the multiplicative property described in section 3, but this is a very promising line of attack which should be pursued further.

Annexe C

Key Recovery on Hidden Monomial Multivariate Schemes

EUROCRYPT 2008
[120] avec Gilles Macario-Rat et Jacques Stern

Abstract : *The problem we study in this paper is the key recovery problem on the C^* schemes and generalizations where the quadratic function of C^* is replaced by a monomial of higher degree. This problem has been further generalized to any multivariate function hidden with two invertible linear maps and named the Isomorphism of Polynomials (IP) problem by Patarin et al. Some cryptosystems have been built on this appearing hard problem such as a traitor tracing scheme proposed by Billet and Gilbert. Here we show that if the hidden multivariate function is a quadratic monomial, as in SFLASH, or a monomial of higher degree as in the traitor tracing scheme, then it is possible to recover an equivalent secret key in polynomial time.*

C.1 Introduction

Multivariate cryptography provides alternative schemes to the RSA cryptosystem where the hard underlying problem consists in solving a system of multivariate equations over a finite field. This problem is known to be NP-complete and there is no known polynomial-time quantum algorithm to solve it. Moreover, generic solvers that use Gröbner basis require exponential time and memory.

One rich family of multivariate scheme is derived from a cryptosystem proposed by Matsumoto and Imai since 1988. Even though this scheme was broken by Patarin in [238] since 1995, Patarin proposed various countermeasures to increase the security. One variation is the Minus transformation, suggested by Shamir in [268], and is a classical solution to avoid Patarin's or Gröbner basis attack. The SFLASH signature scheme comes from this variation. Another scheme solution is to use a hidden monomial of higher degree such as in the Traitor Tracing scheme proposed by Billet and Gilbert in [36].

C.1.1 Related Works

The IP problem.

The Isomorphism of Polynomials (IP) problem has been introduced by Patarin since 1996 in [240] to capture the key recovery problem of some multivariate schemes such as C^* since Patarin's attack allows only to inverse the public key and not to recover the secret key. Patarin, Goubin and Courtois investigate the hardness of this problem in [243] and conclude that the best algorithm, called To and Fro, for the C^* family requires exponential time in $O(q^{n/2})$ where q is the size of the base finite field and n is the degree of the extension of the large field.

Biryukov *et al.* propose another solutions to this problem in [37] with complexity $O(n^3 \cdot 2^n)$ over $GF(2)$ which is very efficient when $n \leq 32$. They introduce these algorithms to study linear equivalence of Sbox. For our purpose, n can be 128. In the case of the AES cryptosystem, the Sbox can be viewed as polynomial of high degree, namely 7, since the inverse in $GF(256)$ can be explained as the polynomial $P(x) = x^{q-2}$.

Finally, Faugère and Perret in [104] also studied this problem and conjecture that in some cases, Gröbner basis algorithms are subexponential and give some parameters that they were able to solve.

Differential Attack on SFLASH.

Recently, some breakthrough results have been published on the cryptanalysis of the SFLASH signature scheme by Dubois *et al.* in [97, 96]. SFLASH comes from the C^* family, *i.e.* the internal quadratic monomial of the form $P(x) = x^{1+q^\theta}$ over an extension $\mathbb{F}$ of degree n of the base finite field $\mathbb{K}$ is hidden by two linear bijective mappings S and T. The public key is $\mathbf{P} = T \circ P \circ S$ and if some polynomials of the public key are removed, we get a SFLASH public key. In[97], the authors consider the case where $\gcd(\theta, n) > 1$.

The basic idea of [137, 97, 96] is to recover some of these polynomials or of equivalent polynomials by noticing that the internal polynomial $P \circ S$ over $\mathbb{F}$ forms a set of n polynomials over $\mathbb{K}$. Then, the action of T consists of linear combination of these n polynomials. Consequently, if we are able to recover other linear combinations of these polynomials with independent coefficients, we will be able to recover a complete public key.

The last results show that it is possible to reconstruct equivalent missing polynomials using only 3 polynomials of the public key. The way to do it is to reconstruct some special linear applications related to the secret S, of the form $N_z = S^{-1} M_z S$ so that M_z denotes the multiplications by z in $\mathbb{F}$. In [97], it is shown that the maps N_z where z are the $q^\theta - 1$ roots of unity are easy to recover using a linear characterization, whereas in [96], more involved analysis are needed. However, this last attack is more powerful since any multiplication can be recovered. Then, the composition of these maps N_z with the public key $\mathbf{P}$ is of the form $T \circ P \circ M_z \circ S$ and since P is multiplicative, $\mathbf{P} \circ N_z$ is of the form $T' \circ P \circ S$ and if T' contains rows independent of those of T, then we get new polynomials of the public key which will be independent from the first ones. Finally, once the public key is recovered, Patarin's attack can be applied.

C.1.2 Our Results

In this paper, we show that the recent attacks on multivariate schemes can be made more devastating and lead to total break of the C^* schemes family. More precisely, we show that the IP problem for C^* is easy and we can recover secret keys S and T or equivalent can be recovered given a $N_z = S^{-1} M_z S$ linear mappings. Indeed, these matrices depend

on the secret S, but M_z are unknown. Here, we show how we can recover z and then, how we can recover S' and T'. This last step is not always easy and when $\gcd(n, \theta) > 1$, many parasitic solutions can happen. For the SFLASH signature scheme, the recent attacks rely on Patarin's attack in their final stage. However, this attack can become exponential in some bad cases. Here, our attack on the C^* schemes family is always polynomial to recover the secret key and can be seen as a new attack on the C^* scheme.

Moreover, we show that for high degree monomials, we can also recover the N_z as in the case of the quadratic polynomials of SFLASH and recover the secret keys. These two results improve on a result of Faugère and Perret at Eurocrypt '06 using Gröbner basis [104] which solves only some particular cases but not all the proposed parameters by Billet and Gilbert. For the C^* case, Faugère and Perret indicate that their approach cannot take into account $n = 19$ and $n = 37$ in SFLASH over a finite field of 2^7 elements. Moreover, for some parameter they define, they conjecture that their attack is subexponential. Here, we only present polynomial time attack to recover these values.

C.1.3 Organization of the Paper

In section 2 we present the problem Isomorphism of Polynomials which represents the key recovery problem in multivariate schemes. Then, we present the differential of the public key which allows to give a characterization of the interesting linear mappings we are looking for. Then, we show how to solve the IP problem when the internal polynomial is a monomial in section 4. In section 5, we show that the SFLASH public key can be recovered in all cases and on monomial of higher degree of the traitor tracing scheme before the conclusion.

C.2 Isomorphisms of Polynomials Problem (IP)

In this section, we present the Isomorphism of Polynomials problem stated by Patarin *et al.* in [240, 243]. It has been used by Billet and Gilbert in [36] to define a traitor tracing scheme.

C.2.1 Description of the IP Problem

Let $\mathbb{K}$ be a small finite field of q elements and $\mathbb{F}$ an extension of degree n over $\mathbb{K}$. Let π be a natural isomorphism from $\mathbb{K}^n$ onto $\mathbb{F}$ and P some polynomial over $\mathbb{F}$. All these elements are supposed to be publicly known. Then, let S and T be two linear or affine invertible transformations over $\mathbb{K}^n$. The maps S and T are kept secret. Finally let $\mathbf{P} = T \circ \pi^{-1} \circ P \circ \pi \circ S$ be a set of n polynomial forms over $\mathbb{K}^n$. This system of multivariate polynomials $\mathbf{P}$ is also named the public key. The problem can now be expressed as follows :

IP Problem.

Given $\mathbb{K}$, $\mathbb{F}$, P, and $\mathbf{P}$ defined as above, find S' and T' affine transformations over $\mathbb{K}^n$ such as :

$$\mathbf{P} = T' \circ \pi^{-1} \circ P \circ \pi \circ S'.$$

In the sequel, by some misuse of language, we avoid writing the isomorphism π and its inverse π^{-1} when their use is obvious and simply write $\mathbf{P} = T \circ P \circ S$.

IP with Polynomials.

In this article, we mainly study the case where P is a monomial, so to say $P(x) = x^d$ for some integer d and we show polynomial time algorithm for these instances. Degree of P and degree of $\mathbf{P}$ are not simply related. Due to the definition of $\mathbb{K}$ and $\mathbb{F}$, we are interested in some special monomials, namely x^{q^i} for some integers i. The views of these monomials on $\mathbb{K}^n$, namely $\pi^{-1} \circ x^{q^i} \circ \pi$ are linear transformations over $\mathbb{K}^n$. If we take for instance $P(x) = x^{1+q^\theta}$ for some integer θ, then $\mathbf{P}$ will be a set of quadratic polynomial forms, hence here $\deg(\mathbf{P}) = 2$. In the same manner, for $P(x) = x^{1+q^{\theta_1}+\ldots+q^{\theta_{d-1}}}$, $\deg(\mathbf{P})$ will be at most d.

C.2.2 Equivalent Keys

Solutions to the IP Problem are in fact not unique. See [296] for a discussion about equivalent keys. For instance, let's analyze the case $P(x) = x^{1+q^\theta}$. Let's note M_z (multiplications) and φ_i (Frobenius) defined by $M_z(x) = zx$ and $\varphi_i(x) = x^{q^i}$. So if (T', S') is a solution then so are

$$(T' \circ \pi^{-1} \circ M_{1/z^{q^\theta+1}} \circ \pi, \pi^{-1} \circ M_z \circ \pi \circ S')$$

and

$$(T' \circ \pi^{-1} \circ (\varphi_i)^{-1} \circ \pi, \pi^{-1} \circ \varphi_i \circ \pi \circ S').$$

C.3 Differential and Properties for Monomials

The differential of the public key of a multivariate scheme has been introduced in a systematic cryptanalytic method by Fouque *et al.* in [110]. Later, this method has been developed and extended in [98, 99, 97, 96] to attack various systems.

C.3.1 Differential of Polynomials

For a general polynomial P, the differential in some point a, denoted by D_aP, is formally defined by :

$$D_aP(x) = P(x + a) - P(x) - P(a) + P(0).$$

We may also refer it as $DP(x, a)$ which is symmetric since $D_aP(x) = D_xP(a)$. The later notation also represents the fact that the differential is a bilinear expression and consequently, it can be represented by a matrix. In our case, all polynomials of the public key can be represented as a bilinear mapping.

The interest of studying the differential is that it "lowers" the degree and it is homogeneous. For instance, if $\deg(\mathbf{P}) = 2$ then $\deg(D_a\mathbf{P}) = 1$ and $D_a\mathbf{P}$ is linear. In this case, the differential acts as it "kills" the parts of degree 1 and 0 of $\mathbf{P}$.

Differential of Monomials of Higher Degree.

For higher degrees, we may define differentials of higher order. For instance, if $\deg(\mathbf{P}) = 3$:

$$D_{a,b}P(x) = D_a(D_bP(x))$$

defines a second order differential and $\deg(D_{a,b}\mathbf{P}(x)) = 1$. We may also note it $DP(a, b, x)$ for the same reason as previously.

Differential of the Public Key.

Let us study how the differential operates on the public key. We assume here that $P(x) = x^{1+q^\theta}$. First, if S and T are linear, then we have

$$D_a\mathbf{P}(x) = T(D_{S(a)}P(S(x))) \tag{C.1}$$

Taking into Account the Affine Parts.

If S and T are affine, we denote by Σ_c the addition with c. With this notation, we have : $(P \circ \Sigma_c)(x) = P(x) + xc^{q^\theta} + x^{q^\theta}c + P(c)$. Now, we can easily express that $D_a(P \circ \Sigma_c)(x) = D_aP(x)$, since $xc^{q^\theta} + x^{q^\theta}c + P(c)$ is affine. Since $S(x) = DS(x) + S(0)$ and $P \circ S = P \circ \Sigma_{S(0)} \circ DS$, we deduce a similar relation :

$$D_a\mathbf{P}(x) = DT(D_{DS(a)}P(DS(x))). \tag{C.2}$$

So, relation (C.2) is just like relation (C.1) where S and T are replaced by their linear part DS and DT.

C.3.2 Multiplicative Property of the Differential

In this section, we show that a characterization equation exists for hidden monomials that involves a linear mapping N. Since the equation is linear in the unknown of N and depends only on the public key, N can be easily found.

Invariants for SFLASH.

For $P(x) = x^{1+q^\theta}$ there is an interesting property of the differential :

$$D_xP(M_z(y)) + D_yP(M_z(x)) = M_{z+z^{q^\theta}}(D_yF(x)) \tag{C.3}$$

where M_z is the multiplication by z in $\mathbb{F}$. We can also rewrite this equation as

$$DP(xz, y) + DP(x, yz) = (z + z^{q^\theta})DP(x, y).$$

How is this property (C.3) transfered to the public system ? Firstly for the sake of simplicity, we may assume that S and T are linear. Otherwise, we will see that considering only their linear part is a good approach when they are affine.

If we denote by N_z the conjugate by S of M_z, namely $N_z = S^{-1} \circ M_z \circ S$, property (C.3) becomes :

$$\begin{aligned} D_x\mathbf{P}(N_z(y)) + D_y\mathbf{P}(N_z(x)) &= T(M_{z+z^{q^\theta}}(D_{S(y)}F(S(x)))) \\ &= (T \circ M_{z+z^{q^\theta}} \circ T^{-1})(D_y\mathbf{P}(x)) \end{aligned} \tag{C.4}$$

Invariants for Monomial of High Degree.

For degree 3 or 4, similar expressions for this property can be derived, by considering respectively :

$$D_{x,y}\mathbf{P}(N_z(u)) + D_{x,u}\mathbf{P}(N_z(y)) + D_{y,u}\mathbf{P}(N_z(x)), \tag{C.5}$$

$$D_{x,y,u}\mathbf{P}(N_z(v)) + D_{x,y,v}\mathbf{P}(N_z(u)) + D_{x,u,v}\mathbf{P}(N_z(y)) + D_{y,u,v}\mathbf{P}(N_z(x))). \tag{C.6}$$

C.4 Recovering S and T

The basic idea to recover an equivalent S and T is to use equation :

$$N_z = S^{-1} M_z S.$$

If we can recover z, then M_z is known and we can linearized it to :

$$SN_z = M_z S$$

where S is the unknown we are looking for.

Description of the Attack.

In the following, we describe the different steps of the attack to recover equivalent S and T.

1. Find all linear transformations L such as $D_x\mathbf{P}(L(y)) + D_y\mathbf{P}(L(x))$ is a set of bilinear forms, all of them being linear combinations of the elements of $D_y\mathbf{P}(x)$.
2. Pick up at random one solution L which characteristic polynomial is not irreducible over $\mathbb{K}$.
3. Find z such as L and M_z are conjugate. Since L and M_z must have the same characteristic polynomial, choose z as any root of the characteristic polynomial of L.
4. Solve the linear system $X.L = M_z.X$ where the unknown X is a linear mapping of $\mathbb{K}^n$.
5. Pick up at random any non trivial solution S.
6. Compute T as $\mathbf{P} \circ S^{-1} \circ P^{-1}$.

Recovering L.

In [97, 96], it is described how the first step of this attack can be mounted since systems in step 1 is overdefined. Consequently, only a few coordinates of $D_y\mathbf{P}(x)$ are sufficient to solve it. This is the same reason why the "Minus" scheme of SFLASH can be defeated even if some public polynomial are removed.

It is also possible to reconstruct S and T even though they are affine. The computations are the same, but we replace $\mathbf{P}$ by $D\mathbf{P}$. At steps 5 and 6, we can find actually the linear parts of S and T, that is DS and DT. Then, using equation :

$$\begin{aligned}(DT)^{-1} \circ D\mathbf{P}(x) &= D(F \circ S)(x) \\ &= (DS(x))^{1+q^\theta} + (DS(x))^{q^\theta} S(0) + DS(x) S(0)^{q^\theta}\end{aligned}$$

replace x by random values, in order to gain enough linear independent equations, all of the form $ay^{q^\theta} + by + c = 0$, and find the solution $S(0)$. Then, compute $T(0) = \mathbf{P}(0) - (DT \circ P \circ S)(0)$.

Recovering z.

Since M_z and L are similar they have the same characteristic and minimal polynomial. Furthermore, z is a root of the minimal polynomial of M_z, since for any polynomial $p \in \mathbb{F}_q[X]$, we have $p(M_z) = M_{p(z)}$. Furthermore, it is also well-known that the roots

of a minimal polynomial are conjugates, since the coefficients of the minimal polynomial belong to $\mathbb{F}_q q$, and for any element α of $\mathbb{F}_q q$, we have $\alpha^{q^i} = \alpha$, thus for the minimal polynomial p of z, $p(z^{q^i}) = p(z)^{q^i} = 0$. The conjugate property stands also for matrices, since $M_z = (\varphi_q^i)^{-1} M_{z^{q^i}} \varphi_q^i$, where $\varphi_q^i(x) = x^{q^i}$ is the ith frobenius map.

So, once L is known, it suffices to select any of the roots of its minimal polynomial as value for z. Other conjugates would lead to equivalent keys.

Equivalent Keys and Space of Solutions.

At step 1, solutions should be a subspace of dimension n, isomorphic to $\mathbb{F}$, since it is the conjugate by S of the space of multiplication matrices. For instance, trivial solutions are diagonal matrices which correspond to elements of $\mathbb{K}$. So at this step we just need to select any matrix corresponding to a multiplication by a primitive element of $\mathbb{F}$. In degenerate cases when $\mu = \gcd(n, \theta)$ is not 1, this space is isomorphic to an intermediate field between $\mathbb{K}$ and $\mathbb{F}$ of degree μ. At step 3, roots of the characteristic polynomial are conjugate, since it is irreducible over $\mathbb{K}$ and its coefficients belong to $\mathbb{K}$. Thus selecting z^{q^i} instead of z is equivalent to multiply the solutions by φ_i. At step 5, solutions can be obtained from a particular one, by multiplying it by any multiplication matrix M_z.

C.5 Applications

The following experimental results have been obtained with an Opteron 850 2.2GHz, with 32 GBytes of Ram. The systems associated with the instance of the problems and their solutions have been generated using the Magma software, version 2.13-15.

If the following tables, t_{gen} is the time for computing the coefficient of the problem, mainly the linear application that gives $D_x\mathbf{P}(L(y)) + D_y\mathbf{P}(L(x))$ for any L, at step 1, t_{sol} is the time for solving the problem, which is basically a linear algebra issue, regarding intersection of subspaces. 's.' and 'm.' denotes respectively second and minute.

C.5.1 SFLASH Signature Scheme

The following results concern a general instance of IP problem of a C*scheme homogeneous of degree 2, so to say we are looking for linear S and T. Nevertheless, this is almost the problem of key recovery for the SFLASH Signature scheme, where some coordinates (equations) are missing, since finding M_z enables to renegerate missing coordinates.

q	d	n	t_{gen}	t_{sol}
2^7	2	37	6 s.	23 s.
2	2	67	55 s.	10 s.
2^7	2	67	60 s.	12 m.

In the cases $\gcd(n, \theta) > 1$, there exists parasitic solutions when searching the solution for equation $XL = M_z X$ for z a $q^{\gcd(n,\theta)} - 1$ of the unity. It is easy to show that if we have two solutions for this equation X and X', then $X'X^{-1}$ commutes with M_z. In this case, the idea is to apply the second attack of [96] to reconstruct a conjugate of multiplication where z does not live in a subgroup.

C.5.2 Traitor Tracing of Billet and Gilbert

Here as above, the results concern a general instance of IP problem of a C*scheme homogeneous, but of degree 3 and 4. The change was in the use of the expressions (C.5), and (C.6).

q	d	n	t_{gen}	t_{sol}
2^8	4	11	18 s.	46 s.
2^9	3	19	15 s.	11 s.

These results confirm experimentally the complexity of the resolution of the problem, namely $\log(q)d^n$. Compare with the results of Faugère and Perret in [104], this is better, since we can handle directly the maximum degree.

C.6 Conclusion

Here, we describe a key recovery attack on the C*schemes family which lead to recover equivalent secret keys. This means that an attacker would be in the same position than a legitimate user. Moreover, this attack is polynomial in time and space, and so it is very pratical and can be executed within few seconds on the recommended values of the parameters of the schemes.

Annexe D

Probabilistic Algorithms for the Equivalence of Quadratic Maps

avec Charles Bouillaguet et Amandine Véber

Abstract : *We give several algorithms that solve the problem of the equivalence of quadratic maps over finite fields. This is a natural generalization of the problem of testing whether two linear maps (or, equivalently, matrices) are equivalent. Geometrically speaking, two linear maps are equivalent if they could be made equal by choosing appropriate bases of both the input and output vector spaces. We consider the same problem, with quadratic maps instead of linear ones. The linear case is fairly easy, since two matrices are equivalent if and only if they have the same rank. In the quadratic case, things are more complex, and the problem is known to be Graph-Isomorphism-hard. This problem has also received some attention from the cryptographic community, and its hardness has been used to build secure public-key encryption, signature or identification schemes.*
We first give and analyze a deterministic algorithm that solves the easy case where the quadratic maps is inhomogeneous, i.e., also has a linear component. We show that it solves a large constant fraction of the instances, and that it terminates in time $\mathcal{O}\left(q \cdot n^6\right)$ with high probability, where q is the size of the field and n is the dimension of the vector space.
We then discuss two randomized algorithms for the hard case where the quadratic map is homogeneous. Both rely on the birthday paradox to improve on the complexity of naive methods. The first one runs in time $\mathcal{O}\left(n^4 q^{2n/3}\right)$, while the second runs in time $\mathcal{O}\left(n^{3.5} q^{n/2}\right)$. The latter works by finding collisions between large sets of random unordered trees constructed from the quadratic maps.

D.1 Introduction

The equivalence of linear maps is a well-known problem where given two matrices A and B, we try to find two invertible matrices T and S such that $B = T \times A \times S$. It is well-known that this problem has a solution if and only if A and B have the same rank, and a possible pair (S, T) can be determined using linear algebra.

In this paper, we consider a natural generalization of this equivalence problem to *quadratic maps*. A quadratic map $f : \mathbb{K}^n \to \mathbb{K}^n$ is essentially a vector of n quadratic poly-

nomials, expressing each coordinate of the output vector as quadratic expressions of the coordinates of the input vector. In other terms, a matrix is a vector of linear polynomials, while a quadratic map is a vector of quadratic polynomials. Given two quadratic maps **a** and **b**, we try to find two invertible matrices such that

$$T \circ \mathbf{b} = \mathbf{a} \circ S \tag{D.1}$$

Geometrically speaking, two equivalent quadratic (resp. linear) maps describe the same transformation, but with coordinates expressed in potentially different bases. This Quadratic Maps Equivalence (QME) problem has been (quite badly) called the "IP problem" (Isomorphism of Polynomials) by Patarin in 1996 [240] and has received some attention from the cryptographic community. During the blossom of "multivariate cryptography", many Trapdoor One-Way Functions were proposed, based on the following idea : the public-key is a quadratic map, and evaluating the TOWF essentially amounts to evaluate the quadratic map. The intuitive security argument is that inverting arbitrary quadratic maps is well-known to be NP-hard over any finite field. The trapdoor is that these quadratic maps have a special structure allowing the owner of the secret key to invert them. Usually, the idea was to choose an easily-invertible quadratic map f as well as two random invertible matrices S and T, and to hope that $f' = T \circ f \circ S$ would be hard to invert without knowledge of S and T (and eventually f). In some of these systems, the "internal map" f was public, and recovering the secret key from the public key could be done by solving an instance of QME. Most of these cryptosystems were broken because the structure of the trapdoor was not hidden well-enough by the linear "masking". In some cases, the secret key could be directly deduced from the public-key by "breaking" special cases of QME.

In general (*i.e.* with randomly generated instances), QME can be difficult, and only exponential algorithms are known for the hard cases. These hard cases can also be used in cryptography to design a zero-knowledge identification scheme : take the identification scheme based on a zero-knowledge proof system for graph-isomorphism of Goldreich, Micali and Widgerson [141], and replace "graph-isomorphism" by "quadratic maps equivalence" in it. Interestingly enough, a special case of QME (where T is the identity matrix) has been shown to be Graph-Isomorphism-hard [243], and the polynomial hierarchy would collapse if QME were NP-complete, just like Graph Isomorphism [243, 104].

Essentially two non-trivial algorithms have been proposed so far to solve QME : the "To-and-Fro" approach of Courtois, Goubin and Patarin [243] on the one hand, and the "Gröbner Basis" approch of FaugÃ¨re and Perret [104] on the other hand. The former uses the fact that knowledge of S can easily be transfered into an exponentially larger knowledge of T by evaluating **a** and **b**. This larger knowledge of T can then be transfered back to S by inverting **a** and **b** about n times. The complexity of this approach is lower-bounded by the cost of these n inversions plus that of finding one or two initial relations on S, but its analysis is incomplete – for instance, the inverses may not exist and the technique may fail, a fact that is not taken into account in [243]. Using exhaustive search to invert **a** and **b** yields a complexity of order $\Omega\left(n \cdot q^{2n}\right)$.

The Gröbner basis approach is based on the observation that (D.1) yields $n^2(n+1)/2$ quadratic equations in the $2n^2$ unknowns coefficients of S and T – a quite highly over-defined system. Moreover, these equations are bi-homogeneous (the coefficients of T only appear linearly while those of S only appear in quadratic monomials). Gröbner basis algorithms are capable of taking advantage of both features, and asymptotic estimates on the degree of regularity of semi-regular systems by Bardet *et al.* [13, 12] would result in the conjecture that the Gröbner basis approach to QME yields a subexponential algorithm

with complexity $\mathcal{O}\left(e^{\log^3 n}\right)$ in both time and space. Another striking feature of this algorithm is that it empirically terminates in polynomial time $\mathcal{O}\left(n^9\right)$ on a whole class of instances (the random inhomogeneous ones). The actual complexity of this technique is however, as of today, vastly unknown.

Our Results.

We present 3 new algorithms for QME. The first one is deterministic and deals with the inhomogeneous case. It works on a constant fraction (larger than 1/2) of all the instances, and when it does not fail explicitly it terminates in time $\mathcal{O}\left(q \cdot n^6\right)$ with probability greater than 1/2. It combines linear algebra and Gröbner bases, and outperforms the "pure Gröbner basis" approach by several orders of magnitude.

The two other algorithms deal with the more difficult homogeneous case. They are probabilistic and rely on the birthday paradox to beat the q^n bound. The main idea is to reduce such instances to random instances of the inhomogeneous case by using a technique that we called *dehomogenization*. A naive reduction works in time q^{2n}. We use birthday paradox technique to improve the reduction. The first one works in time $\mathcal{O}\left(q^{2n/3}\right)$ and is simple to analyze. The last one is a bit more exotic : it works by finding collisions between an exponential number of random unordered trees built from the kernel of the differential of quadratic maps. It has time and space complexity $\mathcal{O}\left(q^{n/2}\right)$ and outperforms all the other techniques in practice. Its analysis makes use of both known and novel results on branching processes and Galton-Watson trees. In order to analyze the complexity of our reduction, we have shown the following theorem. Finally, we provide an as-rigorous-as-possible analysis of our algorithms and we confirmed our theoretical results by experiments.

Main Informal Theorem. *Sampling α^n random critical trees conditioned on having height at least $n \log n$ yields $\alpha^n(1-o(1))$ pairwise non-isomorphic trees with overwhelming probability, for any value of $\alpha > 1$.*

Techniques used.

Our algorithms and their analysis, require the invocation of a variety of mathematical tools. We differentiate quadratic maps, following ideas that have been used to study the security of multivariate scheme [98, 97, 96, 120] and also in the complexity theory to construct pseudorandom generators that fool low-degree polynomials over finite field [206, 42, 43]. The differential of quadratic maps is a matrix and we are interested in the rank or kernel of this matrix. We use classical tools of linear algebra, notably linking the dimension of the commutant of a matrix with its similarity invariants. We make a heavy use of combinatorial properties of vector spaces and of linear maps over finite fields – for instance, how many matrices contain a given vector in their image ? Lastly, we use well-known probabilistic tools such as the birthday paradox and classical results on critical Galton-Watson trees. However, some of the results we needed were not available in the literature : if you sample N random trees according to some fixed distribution, how many of them do you expect to be pairwise non-isomorphic ? We used known tools (the *spine decomposition* [135]) to obtain these results.

Organisation of the paper.

We present some basic ingredients in section D.2. Then, in section D.3, we present and analyze our deterministic algorithms to solve the inhomogeneous QME problem. Finally,

in section D.4, we present two probabilistic algorithms for solving the homogeneous QME problem. Performance results and proofs have been moved to the appendices.

D.2 Preliminaries

Throughout this paper, we consider quadratic polynomials in n variables, *i.e.* elements of the set $\mathbb{F}_q[x_1, \ldots, x_n]_{\leq 2}$. A *quadratic map* is a function from $(\mathbb{F}_q)^n$ to $(\mathbb{F}_q)^n$ such that each coordinate is given by a quadratic polynomial. It is in fact a vector of n quadratic polynomials. We denote by $\mathcal{L}_n$ the set of all endomorphisms of $(\mathbb{F}_q)^n$, by $\mathcal{Q}_n$ the set of all quadratic maps of $(\mathbb{F}_q)^n$, and by $\mathrm{GL}_n(\mathbb{F}_q)$ the set of bijective endomorphisms of $(\mathbb{F}_q)^n$.

Differentials.

Given a quadratic map $f : \mathcal{Q}_n$, its *differential* at a point $c \in (\mathbb{F}_q)^n$ is the *linear map* defined by :

$$\frac{\partial f}{\partial c}(\vec{\mathbf{x}}) = f(\vec{\mathbf{x}} + c) - f(\vec{\mathbf{x}}) - f(c) + f(0)$$

By an abuse of notation, we denote by $\frac{\partial \mathbf{a}}{\partial c}$ the corresponding matrix. It must be noted that if q is even, then $\frac{\partial \mathbf{a}}{\partial c}$ vanishes on c. The interest of considering the differential is that for all $\vec{\mathbf{x}}$:

$$T \circ \mathbf{b} = \mathbf{a} \circ S \Longrightarrow T \times \frac{\partial \mathbf{b}}{\partial \vec{\mathbf{x}}} = \frac{\partial \mathbf{a}}{\partial (S \cdot \vec{\mathbf{x}})} \times S \tag{D.2}$$

Combined with this equation, the following result is one of the main tools for analyzing our algorithms.

Lemme D.1 ([98], theorem 2). *Let $\mathbb{F}_q$ be a field of characteristic two. Given a linear map $L \in \mathcal{L}_n$ and a vector $\vec{\mathbf{x}}$ such that $L(\vec{\mathbf{x}}) = 0$, then the number of quadratic map $f \in \mathcal{Q}_n$, such that $\frac{\partial f}{\partial \vec{\mathbf{x}}} = L$ is independent from L and $\vec{\mathbf{x}}$. Therefore, the differential of a random quadratic map of a point is a random matrix vanishing at that point.* ▽

Homogeneous Components.

Given a polynomial f, we denote by $f^{(k)}$ its *homogeneous component* of degree k, *i.e.*,the sum of all the degree-k terms appearing in f (f is therefore the sum of all its homogeneous components). This also extends naturally to quadratic maps. Observe in particular that $\mathbf{a}^{(1)}$ is also an endomorphism of $(\mathbb{F}_q)^n$, and that $\mathbf{a}^{(0)}$ is a vector of $(\mathbb{F}_q)^n$. Because S and T are linear maps, they transform the homogeneous components of $\mathbf{a}$ into those of $\mathbf{b}$. This has in particular the following consequences :

$$T \times \mathbf{b}^{(1)} = \mathbf{a}^{(1)} \times S \tag{D.3}$$

$$T \times \mathbf{b}^{(0)} = \mathbf{a}^{(0)} \tag{D.4}$$

D.2.1 Finite Vector Spaces Combinatorics

To fully exploit lemma D.1, we need to explore some combinatorial properties of $\mathcal{L}_n$. We found [95] very helpful for this purpose. Let $\vec{\mathbf{x}}$ and $\vec{\mathbf{y}}$ be two linearly independent vectors, and let us consider the following subsets of $\mathcal{L}_n$: $D(n, d)$ denote the matrices having a kernel of dimension d, $K(n, \vec{\mathbf{x}})$ the matrices vanishing on $\vec{\mathbf{x}}$, and $I(n, \vec{\mathbf{x}})$ the matrices whose image contain $\vec{\mathbf{x}}$. Lastly, let $S(n, \vec{\mathbf{x}})$ denote the number of matrices $(U, V) \in {\mathcal{L}_n}^2$ such that $U \cdot \vec{\mathbf{x}} = 0$, V is singular and $U + V$ is singular. A simple geometric reasoning

(choosing appropriate bases) shows that the cardinalities of these sets are independent of the choice of $\vec{\mathbf{x}}$ and $\vec{\mathbf{y}}$. It is straightforward that $|\mathcal{L}_n| = q^{n^2}$ and that $|K(n)| = q^{n(n-1)}$. To deal with the rest, let us introduce $\lambda(n) = \prod_{i=1}^{n}(1-q^{-i})$. The proofs of the following identities are given in annex D.7.

$$|D(n,d)| = \frac{\lambda(n)^2}{\lambda(d)^2 \cdot \lambda(n-d)} \cdot q^{n^2-d^2} \tag{D.5}$$

$$|D(n,d) \cap K(n,\vec{\mathbf{x}})| = \frac{\lambda(n) \cdot \lambda(n-1)}{\lambda(d) \cdot \lambda(d-1) \cdot \lambda(n-d)} \cdot q^{n(n-1)-d(d-1)} \tag{D.6}$$

$$|D(n,d) \cap K(n,\vec{\mathbf{x}}) \cap K(n,\vec{\mathbf{y}})| = \frac{\lambda(n) \cdot \lambda(n-2)}{\lambda(d) \cdot \lambda(d-2) \cdot \lambda(n-d)} \cdot q^{n(n-2)-d(d-2)} \tag{D.7}$$

$$|I(n,\vec{\mathbf{x}})| = (q^n-1) \cdot \sum_{i=0}^{n-1} q^{i^2} \cdot \prod_{j=i+1}^{n-1} \left(q^j-1\right)^2 \tag{D.8}$$

$$|K(n,\vec{\mathbf{x}}) \cap I(n,\vec{\mathbf{y}})| = q^{n(n-1)} - \frac{1}{q^n-1} \cdot |I(n,\vec{\mathbf{x}})| \tag{D.9}$$

$$|D(n,d) \cap K(n,\vec{\mathbf{x}}) \cap I(n,\vec{\mathbf{x}})| = \frac{q^{n-d}-1}{q^n-1} \cdot |D(n,d) \cap K(n,\vec{\mathbf{x}})| \tag{D.10}$$

$$|S(d,\vec{\mathbf{x}})| = q^{2N(N-1)} \cdot \left(1 + \left(q^N-1\right) \cdot (1-\lambda(n-1))^2\right) \tag{D.11}$$

Asymptotic Expansions.

Let $\lambda_\infty(q)$ denote the limit of $\lambda(n)$ when n goes to infinity. Note that a random square matrix of dimension n is invertible with probability $\lambda(n) \geq \lambda_\infty(q) > 0$. The Euler pentagonal numbers theorem states that :

$$\lambda_\infty(q) = \lim_{m \to +\infty} \sum_{i=-m}^{m} (-1)^i \cdot q^{-i(3i+1)/2} \tag{D.12}$$

Considering the partial sums yields a close approximation of $\lambda_\infty(q)$, for any value of q. Let ζ_∞ be the limit of $|I(n,\vec{\mathbf{x}})|/q^{n(n-1)}$. It is clear that $\lambda_\infty(q) \leq \zeta_\infty(q)$, and we conjecture that :

$$\zeta_\infty(q) = \sum_{i=0}^{+\infty} (-1)^i \cdot q^{-i(i+1)/2} \tag{D.13}$$

D.3 The (Easy) Inhomogeneous Case

In this section we describe an algorithm that addresses the case where $\mathbf{a}$ and $\mathbf{b}$ are assumed to be random and inhomogeneous, *i.e.*,all the coefficients of all degrees of $\mathbf{a}$ and $\mathbf{b}$ have been chosen uniformly at random in $\mathbb{F}_q$. The algorithm works over any finite field, however its analysis depends on the parity of q. We will assume that $\mathbb{F}_q$ *is of characteristic two*, *i.e.*,that q is a power of two. The experience of the authors with these problems has lead them to conclude that this makes the analysis trickier. The algorithm to which this section is devoted is shown in fig. 7. It is deterministic, and we show that it succeeds over a constant fraction of the instances (it fails explicitly in the other case). When $q = 2$, the algorithm is capable of dealing with only 61% of all the instances, but this proportion is equivalent to $1 - 1/q$, and goes up to 93% as soon as $q = 16$. When the algorithm succeeds, it returns all the solutions of the problem. We also show that its running time is $\mathcal{O}\left(q \cdot n^6\right)$ over a constant fraction of the instances that it can deal with. The proportion

of these easy instances is 37% when $q = 2$, 76% when $q = 3$ and 86% when $q = 4, \ldots$. We emphasize that this does not mean that the algorithm requires much more time on the remaining instances, and we find that it always terminates quickly in practice. We have implemented it inside MAGMA, and we report on its performance in annex D.6.

Algorithme 7 FindIsomorphism-Inhomogeneous$(\mathbf{a}, \mathbf{b})$

1: $\vec{\mathbf{x}} \leftarrow$ particular solution of $\mathbf{b}^{(1)} \cdot \vec{\mathbf{x}} = \mathbf{b}^{(0)}$. **If** no solution exist **then abort.**
2: $\vec{\mathbf{y}}_0 \leftarrow$ particular solution of $\mathbf{a}^{(1)} \cdot \vec{\mathbf{y}} = \mathbf{a}^{(0)}$
3: **for** $\vec{\mathbf{y}} \in \vec{\mathbf{y}}_0 + \ker \mathbf{a}^{(1)}$ **do**
4: Compute a basis $(T_1, S_1), \ldots, (T_r, S_r)$ of the nullspace of :

$$\mathcal{S} : \quad \begin{cases} T \times \mathbf{b}^{(1)} = \mathbf{a}^{(1)} \times S \\ T \times \dfrac{\partial \mathbf{b}}{\partial \vec{\mathbf{x}}} = \dfrac{\partial \mathbf{a}}{\partial \vec{\mathbf{y}}} \times S \end{cases}$$

5: Write down $S = \sum_{i=1}^{r} z_i \cdot S_i$ and $T = \sum_{i=1}^{r} z_i \cdot T_i$, where the z_i are unknown in $\mathbb{F}_q$.
6: Write down $T \circ \mathbf{b} = \mathbf{a} \circ S$ as a system $\mathcal{S}_{\text{quad}}$ of $n^2 \cdot (n+1)/2$ quadratic equations in $z_1, \ldots, z_r$.
7: Find all the solutions of $\mathcal{S}_{\text{quad}}$.
8: **end for**
9: **return** the corresponding values of (S, T).

Description.

The starting point is the fact that equation (D.3) provides n^2 linear relations between the coefficients of S and those of T. This is not enough to recover the two matrices by linear algebra, since there are $2n^2$ unknowns.

Given a relation $\vec{\mathbf{y}} = S \cdot \vec{\mathbf{x}}$ on S, we would be able to use equation (D.2) to obtain the n^2 missing linear equations. The problem thus comes down to finding such a relation on S. In some cases, this relation can be deduced from equations (D.3) and (D.4) : S transforms the solutions $\vec{\mathbf{x}}$ of the equation $\mathbf{b}^{(0)} = \mathbf{b}^{(1)} \cdot \vec{\mathbf{x}}$ into the solutions of $\mathbf{a}^{(0)} = \mathbf{a}^{(1)} \cdot \vec{\mathbf{y}}$. We see that S transforms an affine space of q^d elements into another subspace of q^d elements, where d is $\dim \ker \mathbf{a}^{(1)}$. So, if $\vec{\mathbf{x}}$ is a preimage of $\mathbf{b}^{(0)}$ through $\mathbf{b}^{(1)}$, then there are q^d possible choices for $\vec{\mathbf{y}}$, and there is –so it seems– no easy apparent way to tell them apart. Therefore, we have to consider all possible cases (this is what the loop of line 4 does).

We will say that an instance of the problem over which the algorithm does not fail on line 2 is *nice*. On a nice instance, then in at least one iteration of the loop we will actually have $\vec{\mathbf{y}} = S \cdot \vec{\mathbf{x}}$. Then, by assembling equations (D.2) and (D.3), we obtain $\mathcal{S}$, a system of simultaneous matrix equivalence. It is made of $2n^2$ homogeneous linear equations in the $2n^2$ coefficients of S and T, and our hope would be that this would be sufficient to retrieve S and T by linear algebra alone – if the kernel of $\mathcal{S}$ contained only the vector (S, T). Unfortunately, our system $\mathcal{S}$ provably contains many parasitic solutions, as lemma D.2 below shows.

Lemme D.2. *If $\mathbf{a}^{(1)}$ is invertible, then the kernel of $\mathcal{S}$ is either trivial, or its dimension is at least n.* ▽

A full proof is given in annex D.8. Its core is that given a particular solution of $\mathcal{S}$, we can build an isomorphism between $\ker \mathcal{S}$ and the commutant of $\mathcal{C} = \left(\mathbf{b}^{(1)}\right)^{-1} \times \frac{\partial \mathbf{b}}{\partial \vec{\mathbf{x}}}$. The

result follows because the commutant of any matrix is of dimension larger than n [27, Fact 2.18.9].

S and T cannot be read off the kernel of $\mathcal{S}$, which has dimension $\Omega(n)$, but our hope is that its dimension is in fact $\mathcal{O}(n)$. We thus compute a basis $\mathcal{B}_{\mathcal{S}} = (S_1, T_1), \ldots, (S_r, T_r)$ of $\ker \mathcal{S}$. This takes time at most $\mathcal{O}(n^6)$, and can very likely be done more efficiently using the block structure of $\mathcal{S}$. Actual solutions of (D.1) can thus be written as linear combinations of r known matrices. Let us therefore introduce r new unknowns $z_1, \ldots, z_r$, and write : $S = \sum_{i=1}^{r} z_i \cdot S_i$ and $T = \sum_{i=1}^{r} z_i \cdot T_i$. The intuition is that this reduces the "entropy" of S and T from $\mathcal{O}(n^2)$ to r field elements. Then, using the already mentioned idea lying at the core of [104], the equation $T \circ \mathbf{b} = \mathbf{a} \circ S$ can be seen as a set $\mathcal{S}_{\text{quad}}$ of $n^2 \cdot (n+1)/2$ quadratic equations in $z_1, \ldots, z_r$. The solutions of $\mathcal{S}_{\text{quad}}$, together with the basis of $\mathcal{S}$, describe all the solutions of (D.1).

Solving systems of quadratic equations is exponential in general (the corresponding decision problem is NP-complete over any finite field [134]). However, if the system is *very overdefined*, then it can be solved efficiently (in polynomial time) assuming that sufficiently many equations are linearly independent from each others. For instance, we could just consider each one of the $\binom{r+2}{2}$ monomials of degree at most two as a new, independent, variable. We would then obtain a system of $n^2 \cdot (n+1)/2$ linear equations in the $\binom{r+2}{2}$ new variables, which is very likely to admit only very few solutions as long as $r \leq n \cdot \sqrt{n}$. In that case, $\mathcal{S}_{\text{quad}}$ can be computed in time $\mathcal{O}(n^6)$ doing only linear algebra.

For the sake of simplicity, in the author's implementation, the equations are simply solved through the computation of a Gröbner basis [58] using the off-the-shelf implementation the algorithm F_4 [105] present in MAGMA. Gröbner basis algorithms terminate very quickly on such overdetermined systems.

Failure Probability.

If an instance is random, then $\mathbf{b}^{(1)}$ is a random linear map, while $\mathbf{b}^{(0)}$ is a random vector. The probability that the latter is non-zero is simply $1 - q^{-n}$. The number of linear maps whose image contains $\mathbf{b}^{(0)}$ is given by (D.8). Thus the probability that a random instance is nice is essentially $P_{\text{nice}} \approx \zeta_\infty(q) = 1 - 1/q + \mathcal{O}(1/q^3)$. When $q = 2$, where this probability is minimal, the algorithm is capable to deal with only 61% of all the instances. This goes up to 93% as soon as $q = 16$.

Expected Number of Iterations of the Loop.

The number of iterations of the loop (taken over the random choice of the instance) is q^d if $\dim \ker \mathbf{b}^{(1)} = d$. Therefore, its expectation is $E = \sum_{d=0}^{n} \lambda(n)^2 / (\lambda(d)^2 \cdot \lambda(n-d))$. We conjecture that $E = 2 - q^{-n}$, which is interesting in terms of complexity. More rigorously, we can easily obtain an interesting deviation bound, by observing that the probability that there will be at most q iterations is :

$$p' = \frac{|K(n,0)| + |K(n,1)|}{q^{n^2}} \leq \lambda_\infty(q) \cdot \left(1 + \frac{q}{(q-1)^2}\right) = 1 - \frac{1}{q^4} + \mathcal{O}\left(\frac{1}{q^5}\right)$$

Rank of $\mathcal{S}$.

Because $\mathbf{b}^{(1)}$ is completely random, it is invertible with probability $\lambda_\infty(q)$, which is about 0.288 for $q = 2$, and growing quickly with q. If it is not, it may become non-singular if we add to it a multiple of $\frac{\partial \mathbf{b}}{\partial \vec{\mathbf{x}}}$. Combining (D.11) and (D.5) shows that if $\mathbf{b}^{(1)}$ is singular,

then $\mathbf{b}^{(1)} + \frac{\partial \mathbf{b}}{\partial \vec{\mathbf{x}}}$ is non-singular with a probability that goes to $\lambda_\infty(q)$ when n grows. All-in-all, we may assume w.l.o.g. that $\mathbf{b}^{(1)}$ is invertible with probability $\lambda_\infty(q)\,(2 - \lambda_\infty(q))$, which is 0.494 for $q = 2$, and grows with q. Then the following result applies :

Lemme D.3. *Let* $p = \mathbb{P}\Big[\dim\ker \mathcal{S} \leq n+2 \;\Big|\; \det \mathbf{b}^{(1)} \neq 0\Big]$,

$$\lim_{n\to\infty} p = \frac{q^5-1}{q^2(q-1)(q^2-1)} \cdot \lambda_\infty(q).$$

A full proof of this lemma is given in annex D.9. The main argument is a known result (of [130]) showing that a random matrix has a minimal polynomial of degree n with non-negligible probability. This result does not directly apply to $\mathcal{C}$, but it can be used to show that $\mathcal{C}$ has a minimal polynomial of degree $n-1$ with non-negligible probability. Then, using another known result linking the dimension of the commutant with the number and the degree of the similarity invariants brings the result.

Summary.

Assembling all the pieces, we find that solving $\mathcal{S}_{\text{quad}}$ can be done in time $\mathcal{O}\left(n^6\right)$ if $\dim\ker \mathcal{S} \leq \sqrt{n}\cdot n - 2$, and by lemma D.3 this happens with a noticeable probability when $\mathbf{b}^{(1)}$ is, or can be made, invertible. This itself happens with probability $\lambda_\infty(q)\cdot(2-\lambda_\infty(q))$. Finally, we expect no more than q iterations of the loop with probability p'. Thus, the fraction of the instances that can be solved in time $\mathcal{O}\left(q \cdot n^6\right)$ is about :

$$\begin{aligned} p'' &= \lambda_\infty(q)^3 \cdot ((2-\lambda_\infty(q)) \cdot \frac{q^5-1}{q^2(q-1)(q^2-1)} \cdot \left(1 + \frac{q}{(q-1)^2}\right) \\ &= 1 - \frac{1}{q^2} - \frac{3}{q^3} - \frac{3}{q^4} + \mathcal{O}\left(\frac{1}{q^5}\right) \end{aligned}$$

The main obstacle in a possible extension of the analysis is that it is difficult to control the rank of $\mathcal{S}$ without assuming that the linear part of the instance is invertible. However, one may expect that if it is of rank, say, $n-1$, then the situation is not very different in practice.

D.4 The Homogeneous Case

Unfortunately, the method discussed in the previous section cannot be applied in the homogeneous case for obvious reasons – we have $\mathbf{a}^{(1)} = \mathbf{b}^{(1)} = 0$. However, it can be used as a sub-component. Since there is no longer a linear homogeneous component to give us n^2 "free" linear relations, our strategy will be to acquire two relations $\vec{\mathbf{y}}_1 = S \cdot \vec{\mathbf{x}}_1$ and $\vec{\mathbf{y}}_2 = S \cdot \vec{\mathbf{x}}_2$, and to consider the following simultaneous matrix equations :

$$\mathcal{S}: \quad \begin{cases} T \times \dfrac{\partial \mathbf{b}}{\partial \vec{\mathbf{x}}_1} = \dfrac{\partial \mathbf{a}}{\partial \vec{\mathbf{y}}_1} \times S \\ T \times \dfrac{\partial \mathbf{b}}{\partial \vec{\mathbf{x}}_2} = \dfrac{\partial \mathbf{a}}{\partial \vec{\mathbf{y}}_2} \times S \end{cases} \tag{D.14}$$

This allows us to carry on as in the inhomogeneous case. Two questions however need to be addressed : How to find the relations on S ? And what is the (expected) dimension of the kernel of $\mathcal{S}$? This last point is a delicate matter because in this case, $\frac{\partial \mathbf{b}}{\partial \vec{\mathbf{x}}_1}$ and $\frac{\partial \mathbf{b}}{\partial \vec{\mathbf{x}}_2}$ define a

singular pencil, and our result based on the dimension of the commutant of a matrix are no longer applicable. We would need to express the dimension of the space containing the solution in this more general case, but we leave this task as a subject of future work. So, we know that we have no formal guarantee that running our inhomogeneous algorithm on systems of the form (D.14) will terminate quickly. In practice, it is very difficult to observe a different behavior on singular instances, as long as the instances are generated randomly, which is good news.

A first and quite naive way to obtain the two relations is to choose $\vec{\mathbf{x}}_1$ and $\vec{\mathbf{x}}_2$ arbitrarily, to enumerate all the possible choices of $\vec{\mathbf{y}}_1$ and $\vec{\mathbf{y}}_2$ and to run the inhomogeneous algorithm on $\mathcal{S}$, and this requires q^{2n} trials. It is possible to do much better by *dehomogenizing* instances.

Dehomogenization

Dehomogenizing a homogeneous instance requires a known relation on S. Indeed, if $\vec{\mathbf{y}}_1 = S \cdot \vec{\mathbf{x}}_1$, then we define $\mathbf{a}'(\vec{\mathbf{x}}) = \mathbf{a}(\vec{\mathbf{x}} + \vec{\mathbf{y}}_1)$ and $\mathbf{b}'(\vec{\mathbf{x}}) = \mathbf{b}(\vec{\mathbf{x}} + \vec{\mathbf{x}}_1)$, and we have $\mathbf{b}' = T \circ \mathbf{a}' \circ S$. If we look at homogeneous components, we find that $\mathbf{a}'^{(2)} = \mathbf{a}^{(2)}$, $\mathbf{a}'^{(1)} = \frac{\partial \mathbf{a}}{\partial \vec{\mathbf{y}}_1}$ and $\mathbf{a}'^{(0)} = \mathbf{a}(\vec{\mathbf{y}}_1)$. Thus, $(\mathbf{a}', \mathbf{b}')$ is an inhomogeneous instance that admits the same solutions as the original problem. However, dehomogenized instances are *not* random inhomogeneous instances, because their linear homogeneous component are always singular. For this reason, dehomogenized instances have a much lower probability P'_{nice} of being nice. We will say that $\vec{\mathbf{x}}$ is *nice with* f if $f(\vec{\mathbf{x}})$ is non-zero and belongs to the image of $\frac{\partial f}{\partial \vec{\mathbf{x}}}$. A dehomogenized instance is nice if $\mathbf{a}$ is nice with $\vec{\mathbf{y}}_1$ or equivalently if $\mathbf{b}$ is nice with $\vec{\mathbf{x}}_1$. We will say that a relation $\vec{\mathbf{y}} = S \cdot \vec{\mathbf{x}}$ is nice if $\mathbf{a}$ is nice with $\vec{\mathbf{y}}$ (or equivalently if $\mathbf{b}$ is nice with $\vec{\mathbf{x}}$). Given a *nice* relation $\vec{\mathbf{y}}_1 = S \cdot \vec{\mathbf{x}}_1$, we can run the inhomogeneous algorithm on the corresponding dehomogenized instance, and we know that it will eventually succeed.

We will assume that $\vec{\mathbf{x}}_1$ being arbitrary and $\mathbf{b}$ being random, $\mathbf{b}(\vec{\mathbf{x}}_1)$ should be non-zero with probability $1 - q^{-n}$. Lemma D.1 tells us that $\frac{\partial \mathbf{b}}{\partial \vec{\mathbf{x}}_1}$ is uniformly random amongst the $q^{n(n-1)}$ linear maps that vanish at $\vec{\mathbf{x}}_1$. Amongst those, $|I(n) \cap K(n)|$ contain $\mathbf{b}(\vec{\mathbf{x}}_1)$ in their image. Combining (D.8), (D.9) and (D.13) shows that the probability that a dehomogenized instance is nice converges quickly to $P'_{\text{nice}} \approx 1 - \zeta_\infty(q)$, and this is $1/q - 1/q^3 + \mathcal{O}\left(1/q^6\right)$. For $q = 2$, where P'_{nice} is maximal, we find $P'_{\text{nice}} \approx 39\%$.

Simple Techniques Based on Dehomogenization.

Armed with the idea of dehomogenizing instances, a simple technique is the following : choose $\vec{\mathbf{x}}_1$ such that $\mathbf{b}$ is nice with $\vec{\mathbf{x}}_1$. Then, enumerate all the possible values of $\vec{\mathbf{y}}_1$ such that $\mathbf{a}$ is nice with $\vec{\mathbf{y}}_1$, dehomogenize and carry on as in the inhomogeneous case. This would require around q^{n-1} invocations of FindIsomorphism-Inhomogeneous, which is already much better than the very naive approach.

An additional improvement is based on the observation that S transforms the zeroes of $\mathbf{a}$ into those of $\mathbf{b}$. If there are k of them, then there are k^2 ways to match them to obtain two relations on S. It is known [131] that a random system of n polynomials in n variables has exactly $s \geq 0$ common zeroes with probability $1/(e \cdot s!)$. Finding the zeroes of $\mathbf{a}$ and $\mathbf{b}$ yields a pair of matrix equation like (D.14) with probability $P'_{\text{nice}}/e + 1 - 2/e$. When $q = 2$, This means that about 40% of the instances are tractable with this method, but this goes down to about 26% when q grows. Even using a very naive approach such as exhaustive search, finding the zeroes of $\mathbf{a}$ and $\mathbf{b}$ is typically faster than running the inhomogeneous algorithm q^n times.

D.4.1 A Rank/Birthday Approach

The previous approach is deterministic, and fails deterministically on a fraction of the instances. We now turn our attention to a probabilistic approach. To improve upon the $\mathcal{O}(q^n)$ bound, a natural idea is to use the birthday paradox. The main idea is that if we build two large-enough random subsets X and Y of $(\mathbb{F}_q)^n$, then there will be with very high probability a pair $(\vec{\mathbf{x}}, \vec{\mathbf{y}}) \in X \times Y$ such that $\vec{\mathbf{y}} = S \cdot \vec{\mathbf{x}}$ (we may even ask that this relation be nice). In that case, we will say that X and Y contain a *right pair*. If X and Y get small enough, then the set of pairs can be tested exhaustively using the inhomogeneous algorithm, and this will succeed if there is a right pair. The birthday "paradox" is that X and Y contain a right pair with probability greater than $1/2$ as soon as they both contain more than $\sqrt{q^n}$ elements, as the following lemma formalizes.

Lemme D.4 ([294]). *N red balls and N blue balls are thrown independently in a uniformly random way into M bins. Let $\mathcal{E}$ be the event that no bin simultaneously contain red and blue balls.*

1. $$\mathbb{P}[\mathcal{E}] = \frac{1}{M^{2N}} \sum_{i=1}^{N} \sum_{j=1}^{N} \left\{ {N \atop i} \right\} \left\{ {N \atop j} \right\} \prod_{k=0}^{i+j-1} (M-k)$$

2. *A first order approximation of the previous expression yields :* $\mathbb{P}[\mathcal{E}] \approx \left(1 - \frac{1}{M}\right)^{N^2}$.▽

Thus, we will have a collision with probability about 63% if we set $N = \sqrt{M}$, and the probability jumps to about 95% if we set $N \approx \sqrt{3M}$. For the sake of obtaining simpler expressions, we shall be content with a success probability of 63%.

To build "small" subsets of $(\mathbb{F}_q)^n$ that have a non-negligible chance of containing a right pair, we use an obvious consequence of (D.2) : if $\vec{\mathbf{y}} = S \cdot \vec{\mathbf{x}}$, then $\frac{\partial \mathbf{b}}{\partial \vec{\mathbf{x}}}$ and $\frac{\partial \mathbf{a}}{\partial \vec{\mathbf{y}}}$ have the same rank. If f is a quadratic map, then let $R_k(f)$ denote the subset of $(\mathbb{F}_q)^n$ formed of points $\vec{\mathbf{x}}$ that are nice with f and such that the kernel of $\frac{\partial f}{\partial \vec{\mathbf{x}}}$ has dimension $k \geq 1$. If the pair $(\mathbf{a}, \mathbf{b})$ has been chosen randomly, then lemma D.1 combined with (D.6) and (D.10) essentially tells us :

$$\mathbb{E}[|R_k(\mathbf{a})|] = \mathbb{E}[|R_k(\mathbf{b})|] = \frac{\lambda(n) \cdot \lambda(n-1)}{\lambda(k) \cdot \lambda(k-1) \cdot \lambda(n-k-1)} \cdot q^{n-k^2} \tag{D.15}$$

Algorithme 8 Function NiceRankList(f, ℓ, k)

```
L ← {}
repeat
    repeat
        x ← random element of (F_q)^n
    until f(x) ≠ 0 and f(x) ∈ Im ∂f/∂x and dim ker ∂f/∂x = k
    L ← L ∪ {x}
until |L| = ℓ
return L
```

The coefficient of q^{n-k^2} is upper-bounded by a small constant, and we may therefore expect $R_k(\mathbf{a})$ to be non-empty if $k \leq \lceil \sqrt{n} \rceil$. Our algorithm is shown in figure **??**, and depends on two parameters ℓ and k. Lemma D.4 tells us that there is a right pair with probability

Algorithme 9 Function FindIsomorphism-Homogeneous($\mathbf{a}, \mathbf{b}$)

$L_{\mathbf{a}} \leftarrow$ Call NiceRankList($\mathbf{a}, \ell, k$)
$L_{\mathbf{b}} \leftarrow$ Call NiceRankList($\mathbf{b}, \ell, k$)
for $(x, y) \in L_{\mathbf{a}} \times L_{\mathbf{b}}$ **do**
    $(\mathbf{a}', \mathbf{b}') \leftarrow$ Call DeHomogenize($\mathbf{b} = T \circ \mathbf{a} \circ S$, assuming that $y = S \cdot x$).
    Call FindIsomorphism-Inhomogeneous($\mathbf{b}' = T \circ \mathbf{a}' \circ S$)
    if solution is found **then**
        Abort the loop, report **success**.
    end if
end for
Report **failure**.

63% if $\ell = \sqrt{|R_k|}$, so we choose $\ell = \sqrt{\mathbb{E}[|R_k|]}$. We then expect q^n/ℓ^2 iterations of the inner loop of NiceRankList to find a good vector. Next, choosing $k = \sqrt{n/3 - 2\log_q n}$ yields $\ell \approx q^{n/3}/n$, so that NiceRankList performs $n \cdot q^{2n/3}$ matrix operations, while there will be $q^{2n/3}$ calls to the inhomogeneous algorithm in the main loop. In each of these calls, there will be about $q^{\sqrt{n/3}}$ iterations of the main loop, because the kernels of the linear components have dimension d.

All in all, the complexity of this approach is $\mathcal{O}\left(q^{2n/3+\sqrt{n/3}}\right)$ which is asymptotically faster than the previous approach. The algorithm has again been implemented inside the MAGMA computer algebra system, and we report on its performance in section D.6. We note that k can only take $\sqrt{n}$ integer values, and that in practice it is difficult to balance the cost of the two parts of the algorithm.

D.4.2 A Branching Process/Birthday Approach.

In the previous approach, we exploited the fact that the rank distribution of the differential is an "invariant" of the equivalence class. However, this invariant splits the (exponentially big) search space in only $\sqrt{n}$ pieces. To improve on the previous approach, we define a more precise invariant in order to partition the search space into exponentially many classes. It follows from (D.2) that if $\vec{\mathbf{y}} = S \cdot \vec{\mathbf{x}}$, then $\ker \frac{\partial \mathbf{b}}{\partial \vec{\mathbf{x}}}$ is transformed into $\ker \frac{\partial \mathbf{a}}{\partial \vec{\mathbf{y}}}$ by S. There is therefore a way to write $\ker \frac{\partial \mathbf{b}}{\partial \vec{\mathbf{x}}} = (v_1, \ldots, v_s)$ and $\ker \frac{\partial \mathbf{a}}{\partial \vec{\mathbf{y}}} = (u_1, \ldots, u_s)$ such that $u_i = S \cdot v_i$. In particular, $\operatorname{rank} \ker \frac{\partial \mathbf{b}}{\partial u_i} = \operatorname{rank} \ker \frac{\partial \mathbf{a}}{\partial v_i}$, so that the following *multiset* equality holds : This recursive definition allows us to associate to each point of $(\mathbb{F}_q)^n$ an unranked, unordered, potentially infinite tree such that if $\vec{\mathbf{y}} = S \cdot \vec{\mathbf{x}}$, then $\textsc{Root}(\mathbf{b}, \vec{\mathbf{x}}) = \textsc{Root}(\mathbf{a}, \vec{\mathbf{y}})$ – the equality is an equality between *unordered* trees, which means that if they are represented as ordered trees, they must be *isomorphic*.

These "rank trees" (ironically, they are "unranked") are more refined invariants than the rank. We focus on their properties, and we derive an improved algorithms based on them in a second step.

Rank Trees.

The number of descendants of a node in these tree is defined by the rank of the differential of $\mathbf{a}$ or $\mathbf{b}$ on some point. Given the random choice of $\mathbf{a}$ and $\mathbf{b}$, it would be reasonable to believe that the number of descendant of each node is randomly sampled according to some distribution. These events are however *not* (even pairwise) independent from each others.

We will nevertheless work under the *heuristic assumption* that these rank trees are random unordered trees where the number of descendant of each node is drawn independently at random according to a specific distribution – in other terms, the rank trees are Galton-Watson trees. We now move on to study their offspring distribution. The number of elements in $\textsc{Leaf}(f, \vec{\mathbf{x}}, \vec{\mathbf{z}})$ depends on the dimension of $\frac{\partial f}{\partial \vec{\mathbf{z}}}$. We know that $\vec{\mathbf{x}} \in \ker \frac{\partial f}{\partial \vec{\mathbf{z}}}$, so we know that this kernel is of dimension at least two. The following extension of lemma D.1 tells us that this is all that we need to know.

Lemme D.5. *Let k be a integer between 2 and n, $\vec{\mathbf{x}}, \vec{\mathbf{y}}$ be linearly independent vectors of $(\mathbb{F}_q)^n$, and $L \in \mathcal{L}_n$ such that $L(\vec{\mathbf{x}}) = L(\vec{\mathbf{y}}) = 0$, with $\dim \ker L = k$. The number of quadratic maps f such that $\frac{\partial f}{\partial \vec{\mathbf{x}}} = L$ is independent from $\vec{\mathbf{x}}, \vec{\mathbf{y}}$ and L.* ▽

Therefore, the rank distribution of $\frac{\partial f}{\partial \vec{\mathbf{z}}}$ follows that of linear maps vanishing on $\vec{\mathbf{z}}$ and $\vec{\mathbf{x}}$. We deduce from (D.7) :

$$p_{n,t} = \mathbb{P}\big[\dim \ker \frac{\partial f}{\partial \vec{\mathbf{z}}} = t\big] = \frac{\lambda(n-2)\lambda(n)}{\lambda(n-t)\lambda(t-2)\lambda(t)} \cdot q^{-t(t-2)}$$

Because of our definition of $\textsc{Root}$ and $\textsc{Leaf}$, the root has a slightly different offspring distribution than the other nodes, but it is not very relevant to our analysis, so we will focus on the non-root nodes. Lemma D.5 tells us that a node has $q^d - q^2$ children with probability $p_{n,d}$. The sums giving the expected number of children μ of a node, as well as its variance σ^2 are not particularly nice, however we conjecture the following identities :

$$\begin{aligned}
\mu &= \sum_{k=2}^{n} \left(q^k - q^2\right) p_{n,k} = 1 - \frac{1}{q^{n-2}}, \\
\sigma^2 &= \sum_{k=2}^{n} \left(q^k - q^2 - \mu\right)^2 p_{n,k} = \left(q^3 - q^2\right)\left(1 - \frac{q^2+1}{q^n} + \frac{1}{q^{2(n-1)}}\right)
\end{aligned}$$

Looking at the (conjectured) expression of μ, we will consider that our Galton-Watson process is critical. [1] It then follows from [10, chapter I, part A, section 5, theorem 1] that these trees are finite with probability one. It is worthwhile to note that the probability that such a random tree has depth greater than k is equivalent to $2/(k\sigma^2)$ [10, chapter I, part B, section 9, theorem 1]. To avoid dealing with too big trees, we truncate the trees we generate to be of height at most h – we will specify this parameter later.

It is known that the expected total number of nodes after h generation is $h+1$ [236], so that computing $\textsc{Root}$ requires on average $\mathcal{O}\left(h \cdot n^3\right)$ operations. The typical tree returned by $\textsc{Root}$ is the tree with only one node, and lemma D.1 combined with (D.6) tells us that this happens with probability $\lambda(n)/\lambda(1) \approx \lambda_\infty(q) \cdot q/(q-1)$, which makes about 0.578 when $q = 2$. Therefore, on most vectors $\vec{\mathbf{x}}$, $\textsc{Root}(f, \vec{\mathbf{x}})$ is relatively uninteresting.

Conditioned Rank Trees.

However, if we restrict our attentions to the subset of $(\mathbb{F}_q)^n$ where $\textsc{Root}(f, \vec{\mathbf{x}})$ *is* interesting, then the situation is quite different. First of all, if we restrict our attention to *high trees*, we may observe more interesting phenomenons. We say that a tree has a *unique spine decomposition* (or *unique backbone decomposition*) if there is a unique path starting from the root that reach a leaf of maximal height. We also say that a tree has a unique spine decomposition *up to height k* if there is a unique path starting from the

1. In fact it is subcritical, because $\mu < 1$, but the difference is so small that it is not perceptible

root and reaching height k that extends to a path reaching nodes of maximal height. The interest of this decomposition is that it allows us to claim that two random trees having a unique spine decomposition up to height k are isomorphic with a probability decaying exponentially fast with k. The probability that a tree conditioned to be of height at least h admits a unique spine decomposition up to height k becomes negligible if k is a bit smaller than h.

More formally, let X_n be a Galton-Watson tree with offspring distribution ℓ_n (here $\ell_n(q^k - q^2) = p_{n,k}$) and let us denote its law by $\mathbb{P}_n$. The average number of offspring of each node is $\mu \approx 1$ (at least when n is large) and its variance σ^2 is finite. For any $n \geq 3$, let $\mathbb{P}_n^H$ be the law of X_n conditioned to have height at least $n \log n$ (w.l.o.g, we write $n \log n$ instead of $\lfloor n \log n \rfloor$, the integer part of $n \log n$).

Théorème D.6. *1. There exists a constant C_1 such that the probability that a random tree sampled according to $\mathbb{P}_n^H$ has a spine decomposition up to height $n\sqrt{\log n}$ is greater than $1 - C_1/\sqrt{\log n}$*

2. There is a constant $\kappa \in]0;1[$ such that if two trees sampled according to $\mathbb{P}_n^H$ have a unique spine decomposition up to height $n\sqrt{\log n}$, then the probability that they are isomorphic is upper-bounded by $\kappa^{n\sqrt{\log n}}$.

3. there exist constants C_1 and C_2 such that if we sample $N'_n = \left(1 - C_2/\sqrt{\log n}\right)^{-1} \cdot N_n$ random trees according to $\mathbb{P}_n^H$, then the probability ε_n that N_n of them a) admit a spine decomposition up to height $n\sqrt{\log n}$ and b) are pairwise non-isomorphic, is lower-bounded by :

$$\varepsilon_n \geq \left(1 - N_n \cdot \kappa^{n\sqrt{\log n}}\right)^{N_n} - e^{-C_1 \cdot N'_n/\sqrt{\log n}} \qquad \diamondsuit$$

Démonstration (sketch). A full proof is given in annex D.10. Theorem D.6 relies on an appropriate criterion to compare two unordered trees and decide whether they are isomorphic or not. The key tool is the *spine* decomposition of a Galton-Watson tree conditioned to have height at least $n \log n$. In essence, it is shown in [135] that such trees have the same law as the random trees constructed as follows : the basis of the tree is a path (the spine) going from the root to height $n \log n$. To the i-th node of the spine, we graft a random number of (unconditioned) i.i.d. Galton-Watson trees with offspring distribution ℓ_i. This measure depends on the height of the node and can be explicitly computed. The important fact here is that it still has finite variance.

Now, the probability that a nearly critical Galton-Watson tree with offspring distribution ℓ_n has height greater than $n \log n - n\sqrt{\log n}$ is bounded by $C/(n \log n - n\sqrt{\log n})$ for some $C > 0$. As a consequence, one can show that with probability at least $1 - C''/\sqrt{\log n}$, none of the trees grafted under height $n\sqrt{\log n}$ attain this height. In this case, only one path from the root reaching height $n\sqrt{\log n}$ extends to a path reaching height $n \log n$, and the spine decomposition is unique up to height $n\sqrt{\log n}$. Because their spines are uniquely identified up to the prescribed height, if two such trees are identical the number of non-trivial subtrees grafted to their spines must be the same at each of the $n\sqrt{\log n}$ levels of interest. However, one can show the existence of $\kappa \in (0,1)$, valid for all n, such that the probability that all subtrees grafted at a given height are empty belongs to $[\kappa, 1-\kappa]$. This result is sufficient to conclude : using the above criterion, the probability that two trees with a unique spine decomposition are equal is bounded by $\kappa^{n\sqrt{\log n}}$. The last point follows from the two others by applying Chernoff's bound and manipulating the result. ∎

A Tree-Based Birthday Algorithm.

Let $H(f)$ be the set of points $\vec{\mathbf{x}}$ of $(\mathbb{F}_q)^n$ that are nice with f, on which $\textsc{Root}(f, \vec{\mathbf{x}})$ has height at least $n \log n$ and admits a spine decomposition up to height $n\sqrt{\log n}$. Given a nice point, a random tree is produced by Root, and it has height at least k with probability about $2/(k \cdot (q^3 - q^2))$. Point i) from theorem D.6 controls the probability of having a spine decomposition. This gives :

$$\mathbb{E}[|H(f)|] = q^{n-5} \cdot \frac{2}{n \log n \cdot (q-1)} \cdot \left(1 - \frac{C_2}{\sqrt{\log n}}\right) \tag{D.16}$$

Algorithme 10 NiceTreeList(f, ℓ)

$L \leftarrow \{\}$
repeat
  repeat
    repeat
      $\vec{\mathbf{x}} \leftarrow$ random element of $(\mathbb{F}_q)^n$
      $T \leftarrow \textsc{Root}(f, \vec{\mathbf{x}})$
    until $f(x) \neq 0$ **and** $f(x) \in \mathrm{Im}\, \frac{\partial f}{\partial \vec{\mathbf{x}}}$
  until T has height at least $n \log n$ **and** T has a unique spine decomposition up to height $n\sqrt{\log n}$
  $L \leftarrow L \cup \{\vec{\mathbf{x}}\}$
until $|L| = \ell$
return L

Algorithme 11 FindIsomorphism-Homogeneous-2$(\mathbf{a}, \mathbf{b})$

$L_{\mathbf{a}} \leftarrow \textsc{NiceTreeList}(\mathbf{a}, \ell, k)$
$L_{\mathbf{a}} \leftarrow \textsc{NiceTreeList}(\mathbf{b}, \ell, k)$
$\mathcal{P} \leftarrow \{(x, y) \in L_{\mathbf{b}} \times L_{\mathbf{a}} \mid \mathrm{CallRoot}(\mathbf{a}, y) = \mathrm{CallRoot}(\mathbf{b}, x)\}$
for $(x, y) \in \mathcal{P}$ **do**
  $(\mathbf{a}', \mathbf{b}') \leftarrow$ Call DeHomogenize($\mathbf{b} = T \circ \mathbf{a} \circ S$, assuming that $y = S \cdot x$).
  Call FindIsomorphism-Inhomogeneous($\mathbf{b}' = T \circ \mathbf{a}' \circ S$)
  if solution is found **then**
    Abort the loop, report **success**.
  end if
end for
Report **failure**.

We adapt the algorithm of the previous section to use rank trees instead of just the rank, and we obtain the pseudo-code shown in algorithm 11. The algorithm will succeed if there is a right pair in $L_{\mathbf{a}} \times L_{\mathbf{b}}$, and again lemma D.4 tells us that setting $\ell = \sqrt{\mathbb{E}[|R_k|]}$ is the way to obtain a success probability of 63%. By point *iii*) of theorem D.6 there will be no more than $(1 + o(1)) \cdot \ell$ iterations of the outer loop of NiceTreeList with overwhelming probability. The complexity of building the lists is therefore $\mathcal{O}\left(n^4 \cdot q^n/\ell\right)$.

Now, we claim that the number of pairs in $\mathcal{P}$ that must be examined in the main function is *constant*. We indeed expect a constant number of right pairs yielding the same trees, but we expect very little "parasitic" solutions : let $X_1, \ldots, X_\ell$ and $Y_1, \ldots, Y_\ell$ be two collections of trees sampled according to $\mathbb{P}_n^H$ and with spine decomposition up to height

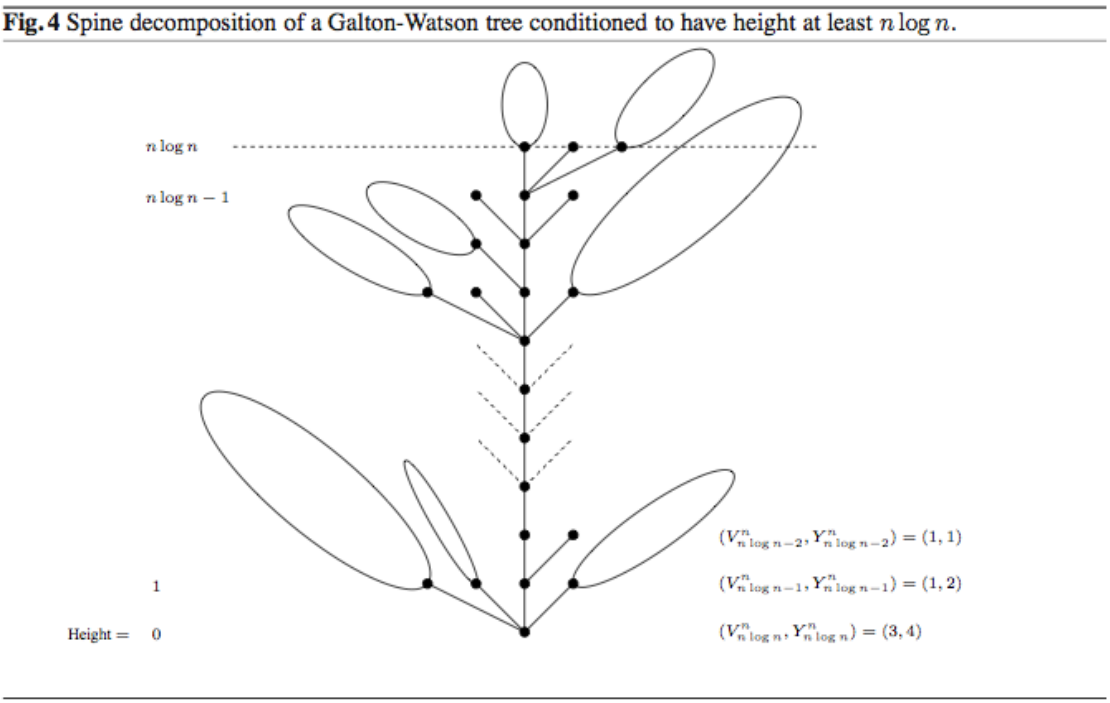

FIGURE D.1 – Spine decomposition of a Galton-Watson tree conditioned to have height at least $n \log n$.

$n\sqrt{\log n}$. The number of collisions between the X's an d the Y's is $\mathcal{N} = \sum_{i,j} \mathbb{1}_{X_i = Y_j}$. If all the X's are different –it is a worst-case assumption–, then applying Markov's bound to $\mathcal{N}$, using point ii) of theorem D.6 and using (D.16) yields :

$$\mathbb{P}[N \geq 1] \leq \sum_{i,j=1}^{\ell} \mathbb{P}[X_i = Y_j] = \ell^2 \cdot \kappa^{n\sqrt{n}} \leq q^n \cdot \kappa^{n\sqrt{n}} \xrightarrow{n \to +\infty} 0$$

The asymptotic complexity of the algorithm is therefore that of generating the lists and finding collisions :

$$n^4 \cdot q^n / \ell \approx n^{3.5} \sqrt{\log n} \cdot q^{n/2+3}$$

D.5 Conclusion

In this paper, we present algorithms for the Quadratic Maps Equivalence problem. One of these algorithms is devoted to the inhomogeneous case and runs in polynomial time with high probability, while the others deal with the harder homogeneous case, and are exponential. Moreover, we explain the complexity, success probability and give sufficient conditions so that the algorithms work. Our techniques for the difficult cases are probabilistic, and combine algebraic and statistical tools.

D.6 Appendix : Implementations and Experimental Results

We implemented the algorithms presented in this paper inside the MAGMA [52] computer algebra system, runnning on one core of a 2.8Ghz Xeon machine. The implementation are not particularly efficient nor optimized, but they enabled us to verify in practice the accuracy of our theoretical estimates.

The inhomogeneous algorithm of section D.3 works very well in practice, and when it succeeds, it works efficiently, as shown in fig. D.2. Comparison with [104] is provided when possible, on comparable hardware.

n	q	generating $\mathcal{S}_{\text{quad}}$	solving $\mathcal{S}_{\text{quad}}$	total time	[104]
20	2^{16}	13s	2s	17s	660s
23	65521	32s	4s	40s	1814s
32	2	120s	24s	148s	
64	2	11400s	1800s	13331s [2]	
32	2^{32}	550s	305s	985s	
16	$2^{521}-1$	16s	8s	30s	

FIGURE D.2: Performance of the inhomogeneous algorithm

The rank-based algorithm of section D.4.1 is more difficult to evaluate, since we typically have very little freedom in the choice of k for values of n where the computation is still tractable. However, we could check in practice that the complexity of building the lists and the expected number of right pairs in them is correct.

n	q	generating $L_{\mathbf{a}}$ and $L_{\mathbf{b}}$	total	k	ℓ	$\lvert\mathcal{P}\rvert$
16	2	0s	68s	3	1	4
22	2	28s	35000s	4	13	400
28	2	4913s	8087s	5	8	64

FIGURE D.3: Performance of the rank-based algorithm

Lastly, the tree-based algorithm of section D.4.1 does work well in practice, as table D.4 shows. It is remarkable that the non-trivial point of the analysis of this algorithm, namely showing that the number of pairs to try is constant, is beautifully confirmed by the experiments. This also justify *a posteriori* the assumption that ROOT returns a random tree.

n	q	generating $L_{\mathbf{a}}$ and $L_{\mathbf{b}}$	computing $\mathcal{P}$	ℓ	$\lvert\mathcal{P}\rvert$
16	2	3.6 s	1s	64	6
24	2	123 s	13s	836	5
32	2	61 min	200s	11585	2
40	2	31 h	2h	165794	7

FIGURE D.4: Performance of the tree-based algorithm

D.7 Appendix : Finite Vector Spaces Combinatorics.

Some of the results and of the methods of this sections are taken from [95]. It is straightforward that $|\mathcal{L}_n| = q^{n^2}$. Let $\vec{\mathbf{x}}$ and $\vec{\mathbf{y}}$ be two non-zero vectors, and let us define the following subclasses of $\mathcal{L}_n$:

$$\begin{aligned}
D_0(k) &= \{f \in \mathcal{L}_n | \dim\ker f = k\} \\
D_1(k,\vec{\mathbf{x}}) &= \{f \in \mathcal{L}_n | \vec{\mathbf{x}} \in \ker f \text{ and } \dim\ker f = k\} \\
D_{1.5}(k,\vec{\mathbf{x}},\vec{\mathbf{y}}) &= \{f \in \mathcal{L}_n | \vec{\mathbf{x}} \in \ker f, \vec{\mathbf{y}} \in \ker f \text{ and } \dim\ker f = k\} \\
D_2 &= \{f \in \mathcal{L}_n | \ker f \subseteq \langle e_2, \ldots, e_n\rangle\} \\
D_{2.5}(k) &= \{f \in \mathcal{L}_n | \ker f \subseteq \langle e_2, \ldots, e_n\rangle \text{ and } \dim\ker f = k\} \\
D_3(\vec{\mathbf{x}},\vec{\mathbf{y}}) &= \{f \in \mathcal{L}_n | \vec{\mathbf{x}} \in \operatorname{Im} f \text{ and } \vec{\mathbf{y}} \in \ker f\} \\
D'_3(\vec{\mathbf{x}},\vec{\mathbf{y}}) &= \{f \in \mathcal{L}_n | \vec{\mathbf{x}} \notin \operatorname{Im} f \text{ and } \vec{\mathbf{y}} \in \ker f\} \\
D_{3.5}(\vec{\mathbf{x}}) &= \{f \in \mathcal{L}_n | \vec{\mathbf{x}} \in \operatorname{Im} f\} \\
D_4(k,\vec{\mathbf{x}},\vec{\mathbf{y}}) &= \{f \in \mathcal{L}_n | \vec{\mathbf{x}} \in \operatorname{Im} f, \vec{\mathbf{y}} \in \ker f \text{ and } \dim\ker f = k\} \\
D'_4(k,\vec{\mathbf{x}},\vec{\mathbf{y}}) &= \{f \in \mathcal{L}_n | \vec{\mathbf{x}} \notin \operatorname{Im} f, \vec{\mathbf{y}} \in \ker f \text{ and } \dim\ker f = k\}
\end{aligned}$$

$D_{3.5}$ apparait section 3, puis D_0, puis D_3 dans 4.1, puis D_4 dans 4.3, puis $D_{1.5}$ dans 4.4, puis D_1 dans 4.4

A simple geometric reasoning (changing bases) shows that the cardinalities of these sets are independent of the choice of $\vec{\mathbf{x}}$ and $\vec{\mathbf{y}}$. We are therefore interested in the expression $D_i^*(k)$ of the cardinality of $D_i(k,\vec{\mathbf{x}},\vec{\mathbf{y}})$.

Let us introduce $\lambda(n) = \prod_{i=1}^{n}(1-q^{-i})$, and let $\lambda_\infty(q)$ denote the limit of $\lambda(n)$ when n goes to infinity. The Euler pentagonal numbers theorem states that :

$$\lambda_\infty(q) = \lim_{m\to+\infty} \sum_{i=-m}^{m} (-1)^i \cdot q^{-i(3i+1)/2}$$

Considering the partial sums yields a close approximation of $\lambda_\infty(q)$, for any value of q, even with $m = 2$. Let $S(n,d)$ denote the number of linearly independent sequences of d vectors of $(\mathbb{F}_q)^n$. It is easily seen that $S(n,d) = \prod_{i=0}^{d-1}(q^n - q^i)$. Each such sequence generates a subspace of dimension d of $(\mathbb{F}_q)^n$, which is also generated by $S(d,d)$ other linearly independent sequences of d vectors. Therefore, the number $E(n,d)$ of d-dimensional subspaces of $(\mathbb{F}_q)^n$ is $S(n,d)/S(d,d)$. We find :

$$S(n,d) = \frac{\lambda(n)}{\lambda(n-d)} \cdot q^{nd} \qquad \text{and} \qquad E(n,d) = \frac{\lambda(n)}{\lambda(n-d)\cdot\lambda(d)} \cdot q^{d(n-d)}$$

D_0^*.

A sequence of r linearly independent vectors from $(\mathbb{F}_q)^n$ uniquely define an injective linear map from $(\mathbb{F}_q)^r$ to $(\mathbb{F}_q)^n$, thus there are exactly $S(n,r)$ of them. Let U be a subspace of $(\mathbb{F}_q)^n$ of dimension d, and V a complementary subspace. Any endomorphism of $(\mathbb{F}_q)^n$ is uniquely defined by its restriction to U and V. An endomorphism admits U as its kernel if and only if it vanishes on U and is injective on V. Thus, there are $S(n,n-d)$ injective linear maps from V to $(\mathbb{F}_q)^n$. And since there are $E(n,d)$ subspaces U of dimension d, we obtain :

$$D_0^*(d) = E(n,d)\cdot S(n,n-d) = \frac{\lambda(n)^2}{\lambda(d)^2\cdot\lambda(n-d)} \cdot q^{n^2-d^2} \tag{D.17}$$

Note that an immediate consequence is that a random square matrix of dimension n is invertible with probability $\lambda(n) \geq \lambda_\infty(q)$.

D_1^*.

Next, the number of subspaces of $(\mathbb{F}_q)^n$ of dimension d and containing $\vec{\mathbf{x}}$ is equal to the number of subspaces of dimension $d-1$ of the quotient $(\mathbb{F}_q)^n / (\vec{\mathbf{x}}.\mathbb{F}_q)$, and this is $E(n-1, d-1)$. Combined with the previous reasoning, this shows :

$$D_1^*(d) = E(n-1,d-1) \cdot S(n, n-d) = \frac{\lambda(n) \cdot \lambda(n-1)}{\lambda(d) \cdot \lambda(d-1) \cdot \lambda(n-d)} \cdot q^{n(n-1)-d(d-1)} \quad \text{(D.18)}$$

D_2^*.

First of all, we fix the image of f on e_1. Since we cannot have $f(e_1) = 0$, there are $q^n - 1$ choices. With an appropriate change of coordinates, we can assume w.l.o.g. that $f(e_1) = e_1$, such that the matrix representation of f is :

$$M = \left(\begin{array}{c|c} 1 & A \\ \hline 0 & B \end{array}\right)$$

Let $\vec{\mathbf{x}} = (x_1 \dots x_n)$ be a vector of $\ker M$ and $\vec{\mathbf{x}}' = (x_2 \dots x_n)$ the corresponding subvector of $\vec{\mathbf{x}}$. We have $x_1 = A \cdot \vec{\mathbf{x}}'$ and $B \cdot \vec{\mathbf{x}}' = 0$. The condition that $\ker M \subseteq \langle e_2, \dots, e_n \rangle$ imposes that $x_1 = 0$, and therefore that $\ker B \subseteq \ker A$. There are two cases : either $A = 0$, and B can be arbitrary, or $A \neq 0$, and $\ker A$ is of dimension $n-2$. In the second case, there are $q^{n-1} - 1$ possible choices of A. We can also assume w.l.o.g. that $\ker A = \langle e_3, \dots, e_n \rangle$ with an appropriate change of coordinates. The set of possible B is therefore the set of endomorphisms of $\langle e_2, \dots, e_n \rangle$ with a kernel contained in $\langle e_3, \dots, e_n \rangle$. Thus we have established the recurrence relation :

$$D_2^*(n) = (q^n - 1) \cdot \left(q^{(n-1)^2} + \left(q^{n-1} - 1 \right) \cdot D_2^*(n-1) \right) \quad \text{(D.19)}$$

Unfolding the recurrence and changing indices yields (the approximation only holds if $n \geq 9$) :

$$D_2^*(n) = (q^n - 1) \sum_{i=0}^{n-1} q^{i^2} \cdot \prod_{j=i+1}^{n-1} \left(q^j - 1 \right)^2 \quad \text{(D.20)}$$

It is straightforward that $D_2^*(n) \cdot q^{-(n^2)}$ goes to a finite limit when n goes to $+\infty$ (because $D_2^*(n) \cdot q^{-(n^2)}$ is bounded and non-decreasing). Let us denote this limit by $\zeta_\infty(q)$. Because $D_0^* \leq D_2^*$, we find that $\lambda_\infty(q) \leq \zeta_\infty(q)$, so that $\zeta_\infty(q)$ quickly goes to 1 when q grows. By observing the first order terms, we conjecture that :

$$\zeta_\infty(q) = \sum_{i=0}^{+\infty} (-1)^i \cdot q^{-i(i+1)/2}$$

$D_{2.5}^*$.

The reasoning is much simpler. The kernel of M is an arbitrary subspace of dimension d of $\langle e_2, \dots, e_n \rangle$ – there are $E(n-1, d)$ of those. Once the kernel of M is fixed, then there are $S(n, n-d)$ ways to choose its image. Thus :

$$D_{2.5}^*(d) = E(n-1,d) \cdot S(n,n-d) = \frac{q^{n-d}-1}{q^n-1} E(n,n-d) \cdot S(n,n-d) = \frac{q^{n-d}-1}{q^n-1} \cdot D_0^*(d)$$

D_3^* and $D_3'^*$.

It is then clear that $D_3^* = q^{n(n-1)} - D_3'^*$. We therefore focus on D_3'. We choose two bases of $(\mathbb{F}_q)^n$, $\mathcal{B}_1 = (\vec{\mathbf{x}}, \vec{\mathbf{x}}_2 \ldots, \vec{\mathbf{x}}_n)$, and $\mathcal{B}_2 = (\vec{\mathbf{y}}, \vec{\mathbf{y}}_2 \ldots, \vec{\mathbf{y}}_n)$, and we consider the matrix representation of an endomorphism meeting our requirements, with the input coordinates expressed in $\mathcal{B}_2$ and expressing the output coordinates in $\mathcal{B}_1$. Then if it vanishes on $\vec{\mathbf{x}}$, M necessarily has the following shape :

$$M = \left(\begin{array}{c|c} 0 & A \\ \hline 0 & B \end{array}\right)$$

In order to have $\vec{\mathbf{x}} \notin \operatorname{Im} M$, we can either have $A = 0$, or have $A \neq 0$ and $\ker B \subseteq \ker A$. If $A = 0$, then B can be arbitrary and this yields $q^{(n-1)^2}$ choices. If A is non-zero –there are $q^{n-1} - 1$ choices–, then its kernel has dimension $n-2$. Up to an appropriate change of coordinates, the set of possible B's in this case is $D_2(n-1)$. Thus we have found :

$$\begin{aligned} D_3'^* &= q^{(n-1)^2} + \left(q^{n-1} - 1\right) \cdot D_2^*(n-1) = \frac{1}{q^n - 1} \cdot D_2^*(n) \\ D_3^* &= q^{n(n-1)} - \frac{1}{q^n - 1} \cdot D_2^*(n) \end{aligned} \tag{D.21}$$

$D_{3.5}^*$.

We choose $(\vec{\mathbf{x}}, x_2, \ldots, x_n)$ as a basis of the output space. Then the first line of M must be non-zero, and we may assume (up to a change of coordinates) that the top-left coefficient is one. This means that $f(e_1) \neq 0$. Let us fix $f(e_1)$ again. Then, up to a suitable change of coordinate, we are exactly in the same situation as we were when studying D_2^*, and with the same constraints. Therefore, $D_{3.5}^* = D_2^*$.

D_4^*.

First, it is obvious that $D_4^*(k) = D_1^*(k) - D_4'^*(k)$. We again focus on D_4'. The matrix can be assumed to be in the same shape as before. Again, we discuss two cases : either $A = 0$, and the only constraint on B is that $\ker B$ must be of dimension $d-1$. Then, there are $D_0(n-1, d-1)$ ways to choose B. Or $A \neq 0$ –there are then $q^{n-1} - 1$ choices for A–, and B must be such that its kernel is of dimension $d-1$, and is included in $\ker A$, which is a subspace of dimension $n-2$. Again, up to a suitable change of coordinate, we may assume that $\ker A = \langle e_3, \ldots, e_n \rangle$. Then the number of possible B's is clearly $D_{2.5}'(n-1, d-1)$. We have obtained :

$$D_4'^*(d) = D_0^*(n-1, d-1) + \left(q^{n-1} - 1\right) \cdot D_{2.5}^*(n-1, d-1)$$

and :

$$\begin{aligned} D_4^*(d) &= D_1^*(n,d) - D_0^*(n-1, d-1) - \left(q^{n-1} - 1\right) \cdot D_{2.5}^*(n-1, d-1) \\ &= D_1^*(n,d) - q^{n-d} \cdot D_0^*(n-1, d-1) \end{aligned}$$

and with a bit of work we find :

$$D_4^*(d) = \frac{q^{n-d} - 1}{q^n - 1} \cdot D_1^*(n,d) = \frac{\lambda(n-1)^2}{\lambda(d) \cdot \lambda(d-1) \cdot \lambda(n-d-1)} q^{n^2 - d^2 - n} \tag{D.22}$$

D.8 Appendix : Proof of Lemma D.2

Let us exclude the case where $\mathcal{S}$ admits no solution, and let us show that $\dim\ker\mathcal{S} \geq n$. We can therefore assume the existence of a particular solution (S_0, T_0). This also guarantees that $\mathbf{b}^{(1)}$ is invertible. Thanks to this, we can write :

$$\mathcal{S}: \quad \begin{cases} T &= \mathbf{a}^{(1)} \times S \times \left(\mathbf{b}^{(1)}\right)^{-1} \\ S \times \left(\mathbf{b}^{(1)}\right)^{-1} \times \frac{\partial \mathbf{b}}{\partial \vec{\mathbf{x}}} &= \left(\mathbf{a}^{(1)}\right)^{-1} \times \frac{\partial \mathbf{a}}{\partial \vec{\mathbf{y}}} \times S \end{cases}$$

Let us define $\mathcal{C} = \left(\mathbf{b}^{(1)}\right)^{-1} \times \frac{\partial \mathbf{b}}{\partial \vec{\mathbf{x}}}$. Note that the second equation can be rewritten :

$$\begin{aligned} S \times \mathcal{C} &= \left(T_0^{-1} \times \mathbf{b}^{(1)} \times S_0^{-1}\right)^{-1} \times \left(T_0^{-1} \times \frac{\partial \mathbf{b}}{\partial \vec{\mathbf{x}}} \times S_0^{-1}\right) \times S \\ &= S_0 \times \left(\mathbf{b}^{(1)}\right)^{-1} \times \frac{\partial \mathbf{b}}{\partial \vec{\mathbf{x}}} \times S_0^{-1} \times S \\ &= S_0 \times \mathcal{C} \times S_0^{-1} \times S \end{aligned}$$

In other terms, $\mathcal{S}$ is equivalent to :

$$\mathcal{S}: \quad \begin{cases} T &= \mathbf{a}^{(1)} \times S \times \left(\mathbf{b}^{(1)}\right)^{-1} \\ \mathcal{C} \cdot \left(S_0^{-1} \cdot S\right) &= \left(S_0^{-1} \cdot S\right) \cdot \mathcal{C} \end{cases}$$

From there, it is not difficult to see that the kernel of $\mathcal{S}$ is in one-to-one correspondance with the commutant of $\mathcal{C}$, the isomorphism being $(S, T) \mapsto S_0^{-1} \cdot S$. The result then follows from the well-known fact that n lower-bounds the dimension of the commutant of any endomorphism on a vector space of dimension n (see for instance [27, Fact 2.18.9]).

D.9 Appendix : Proof of Lemma D.3

The proof of lemma D.2 tells us that if $\mathbf{b}^{(1)}$ is invertible, then the dimension of $\ker\mathcal{S}$ is the dimension of the commutant of $\mathcal{C} = \left(\mathbf{b}^{(1)}\right)^{-1} \times \frac{\partial \mathbf{b}}{\partial \vec{\mathbf{x}}}$. It is known that the dimension of the commutant of a matrix M is $\sum_{j=1}^{s} (2s - 2j + 1) \cdot \deg P_j$, where the P_j are the *similarity invariants* of MThe dimension of the kernel of $\mathcal{S}$ thus only depends on the similarity invariants of $\mathcal{C}$.

Let $P \in \mathrm{GL}_n(\mathbb{F}_q)$ be such that $P \cdot e_1 = \vec{\mathbf{x}}$, and define $\mathcal{C}' = P^{-1} \times \mathcal{C} \times P$. By definition, $\mathcal{C}$ and $\mathcal{C}'$ have the same similarity invariants, thus $\dim\ker\mathcal{S}$ is a function of $\mathcal{C}'$. Next, we claim that $\mathcal{C}'$ is uniformly distributed amongst the matrices that have a first null column. Lemma D.1 tells us that $\frac{\partial \mathbf{b}}{\partial \vec{\mathbf{x}}}$ is uniformly distributed amongst the matrices vanishing at $\vec{\mathbf{x}}$. Because $\left(\mathbf{b}^{(1)}\right)^{-1}$ is a bijection, then the product $\mathcal{C}$ is uniformly distributed amongst the matrices that vanish at $\vec{\mathbf{x}}$. The conjugation by P is a permutation of the set of matrices that transforms matrices vanishing on $\vec{\mathbf{x}}$ into matrices with a null first column.

Let U be the $(n-1) \times (n-1)$ submatrix of $\mathcal{C}'$ obtained by removing the first row and the first column. It is clear from the previous point that U is a random matrix. Also, it is easy to see that $\mathrm{CharPoly}(\mathcal{C}) = X \cdot \mathrm{CharPoly}(U)$, and $\mathrm{MinPoly}(U)$ divides $\mathrm{MinPoly}(\mathcal{C})$. The following lemma is then applicable to U.

Lemme D.7 ([130], theorem 1). *Let $c(n,q)$ be the proportion of cyclic $n \times n$ matrices (i.e.,matrices for which the minimal polynomial is of degree n). We have :*

$$\frac{1}{q^2(q+1)} < 1 - c(n,q) < \frac{1}{(q^2-1)(q-1)} \quad \textit{and} \quad \lim_{n\to\infty} c(n,q) = \frac{q^5-1}{q^2(q-1)(q^2-1)} \cdot \lambda_\infty(q)$$

∇

Lemma D.7 tells us that the probability that the minimal polynomial of U has degree $n-1$, and therefore that the the degree of $\mathrm{MinPoly}(\mathcal{C})$ is at least $n-1$, is $c(n,q)$. Assuming that $\deg \mathrm{MinPoly}(\mathcal{C}) \geq n-1$, there are two possible cases. Either $\mathcal{C}$ is cyclic, and it has one similarity invariant, $\mathrm{MinPoly}(\mathcal{C})$, which is of degree n. In this case, $\dim\ker \mathcal{S} = n$. Or $\mathcal{C}$ has two similarity invariants : X, and $\mathrm{MinPoly}(\mathcal{C})$, which is of degree $n-1$, and $\dim\ker \mathcal{S} = n+2$.

D.10 Proof of Theorem D.6

We need is a criterion to decide whether two (topological) trees are different or not. To this end, we use a decomposition of the conditioned tree into a *backbone* (or *spine*) going from the root to height $n\log n$, on which we graft a given number of unconditioned Galton-Watson trees at each of its nodes. Looking at all nodes of height $n\sqrt{\log n}$, if only one of them has descendants at height $n\log n$ then the spine up to height $n\sqrt{\log n}$ is uniquely determined : necessarily, it is the path in the tree going from the root to this node. The last step consists in comparing the number of empty trees emanating from this portion of the backbone. Indeed, if the spine decomposition is unique up to height $n\sqrt{\log n}$ for both of the trees we are comparing, the numbers of empty subtrees along their spines need to be the same if our two Galton-Watson trees are identical.

To complete the first point, let us fix $n \geq 3$ and work for a moment with ordered Galton-Watson trees. That is, we also record who is the descendant of each parent and offspring are ordered (so that we can talk about brothers to the left or to the right of an individual). Let us denote the ordered Galton-Watson tree with offspring distribution ℓ_n by T_n. For simplicity, we also write $\mathbb{P}_n$ for its law, and $\mathbb{P}_n^H$ for $\mathbb{P}_n[\,\cdot\,|\,H(T_n) \geq n\log n]$. In [135], Geiger shows that if we define the sequence of independent random variables $\{(V_m^n, Y_m^n), m \in \mathbb{N}\}$ by

$$\mathbb{P}[V_m^n = j, Y_m^n = k] = \frac{\mathbb{P}_n[H(T_n) \geq m-1]}{\mathbb{P}_n[H(T_n) \geq m]}\, \mathbb{P}_n[H(T_n) < m-1]^{j-1}\, \ell_n(\{k\}), \qquad 1 \leq j \leq k < \infty,$$

then T_n conditioned to have height at least $n\log n$ has the same law as the random tree constructed inductively as follows :

- The root (i.e., the first node of the spine) has $Y^n_{n\log n}$ offspring.
- To each of the $V^n_{n\log n} - 1$ first offspring node we graft a Galton-Watson tree with offspring distribution ℓ_n and conditioned to have height (strictly) less than $n\log n - 1$. These $V^n_{n\log n} - 1$ trees are independent of each other (and of the rest of the construction).
- To each of the $Y^n_{n\log n} - V^n_{n\log n}$ last offspring, we graft an unconditioned Galton-Watson tree with offspring distribution ℓ_n (again, these trees are independent of each other and of the rest of the construction).
- The $V^n_{n\log n}$-th offspring node continues the spine. It has $Y^n_{n\log n-1}$ offspring, the first $V^n_{n\log n-1}$ ones are the roots of i.i.d. Galton-Watson trees conditioned to have height less than $n\log n - 2$, the last $Y^n_{n\log n-1} - V^n_{n\log n-1}$ are the roots of i.i.d. unconditioned

Galton-Watson trees and the spine carries on with the $V^n_{n\log n-1}$-th offspring, which has $Y^n_{n\log n-2}$ offspring nodes, and so on.

Figure D.1 shows an example of such a decomposition. Observe that the marginal distribution of Y^n_m is given by

$$\mathbb{P}[Y^n_m = y] = \frac{1-\mathbb{P}_n[H(T_n)<m-1]^y}{\mathbb{P}_n[H(T_n)\geq m]} \times \ell_n(\{y\}), \tag{D.23}$$

which is 0 if $y=0$ or if y is not of the form q^k-q^2 for some $k\in\{3,\dots,n\}$. Hence, the spine can be seen as a 'prolific' line of descent that survives up to generation $n\log n$ by producing a biased number of offspring, while the other individuals of the population reproduce essentially according to the initial offspring distribution (we refer to [135] for an explanation of the fact that trees emanating from brothers to the left of the spine are conditioned not to have descendants at generation $n\log n$).

Let us now use this spine decomposition to show that under $\mathbb{P}^H_n$, with high probability only one path from the root to height $n\sqrt{\log n}$ extends to a path reaching height $n\log n$. Call this event A_n. Since this property is purely topological, the same result will then hold for the unordered tree X_n. We obtain the desired result by bounding from below the probability of A_n by the probability that all trees emanating from the spine under height $n\sqrt{\log n}$ are of height less than $n(\log n-\sqrt{\log n})$. The independence of this family of trees, together with the fact (easy to check) that for every $i\in\{1,\dots,n\sqrt{\log n}-1\}$

$$\mathbb{P}_n\big[H(T_n)<n(\log n-\sqrt{\log n})\,\big|\,H(T_n)<n\log n-i\big]\geq\mathbb{P}_n\big[H(T_n)<n(\log n-\sqrt{\log n})\big],$$

enables us to write

$$\begin{aligned}\mathbb{P}^H_n[A_n] &\geq \prod_{i=0}^{n\sqrt{\log n}-1}\mathbb{E}\Big[\mathbb{P}_n\big[H(T_n)<n(\log n-\sqrt{\log n})\big]^{Y^n_{n\log n-i}-1}\Big]\\ &\geq \mathbb{E}\Big[\mathbb{P}_n\big[H(T_n)<n(\log n-\sqrt{\log n})\big]^{\sum_{i=0}^{n\sqrt{\log n}-1}Y^n_{n\log n-i}}\Big].\end{aligned}\tag{D.24}$$

Now, as $n\to\infty$ all $p_{n,k}$, $k\in\{2,\dots,n\}$, converge to a finite limit $p_{\infty,k}$, and we also have $\mu_n\to 1$ (recall $\mu_n<1$ for every n) and $\sigma_n^2\to q^3-q^2=:\sigma^2$. The last two convergences happen exponentially fast in n, therefore the same proof as that of Theorem 3.1 in [135] (in which $\mu_n=1$ for all n) shows that whenever $(m_n)_{n\geq 1}$ tends to infinity at most polynomially, we have

$$\lim_{n\to\infty} m_n\,\mathbb{P}_n\big[H(T_n)\geq m_n\big]=\frac{2}{\sigma^2}. \tag{D.25}$$

Furthermore, we have the following lemma.

Lemme D.8. *There exist $C_3, C_4>0$ such that for every $n\geq 3$,*

$$\mathbb{P}\Big[\sum_{i=0}^{n\sqrt{\log n}-1}Y^n_{n\log n-i}>C_3 n\sqrt{\log n}\Big]\leq\frac{C_4}{n\sqrt{\log n}}.$$

We postpone the proof of Lemma D.8 until the end of the proof of Theorem D.6. Armed with (D.25) and Lemma D.8, we can come back to (D.24) and write for every n

$$\begin{aligned}\mathbb{P}^H_n[A_n] &\geq \mathbb{E}\Big[\mathbb{P}_n\big[H(T_n)<n(\log n-\sqrt{\log n})\big]^{C_3 n\sqrt{\log n}}\mathbf{1}_{\left\{\sum_{i=0}^{n\sqrt{\log n}-1}Y^n_{n\log n-i}\leq C_3 n\sqrt{\log n}\right\}}\Big]\\ &\geq \left(1-\frac{C_5}{\sigma^2 n\log n}\right)^{C_3 n\sqrt{\log n}}\times\mathbb{P}\Big[\sum_{i=0}^{n\sqrt{\log n}-1}Y^n_{n\log n-i}\leq C_3 n\sqrt{\log n}\Big]\\ &\geq e^{-C_6/\sqrt{\log n}}\left(1-\frac{C_4}{n\sqrt{\log n}}\right)\geq 1-\frac{C_7}{\sqrt{\log n}}.\end{aligned}\tag{D.26}$$

(For the third inequality, use the fact that $1-x \geq e^{-2x}$ for every $x \in [0,1/2]$.)

Let us conclude our first step by observing that the spine is unique up to height $n\sqrt{\log n}$ in the ordered Galton-Watson tree T_n iff it is unique in the corresponding topological tree. This is easy to see from the formulation of the event A_n in terms of paths of length $n\sqrt{\log n}$ extending to a path of length $n \log n$). As a consequence, what (D.26) shows is that for every $n \geq 3$, if we sample a Galton-Watson tree X_n according to $\mathbb{P}_n^H$, then with probability at least $1-C_7/\sqrt{\log n}$ there will be a unique spine decomposition under height $n\sqrt{\log n}$. Elaborating on this result, we now claim that if $X_n^{(1)}, \dots, X_n^{(N_n)}$ are N_n i.i.d. Galton-Watson trees with law $\mathbb{P}_n^H$ and if we call U_n the number of those trees which have a unique spine decomposition under height $n\sqrt{\log n}$, there exist $C_1, C_2 > 0$ satisfying for every n

$$\mathbb{P}_n^H\left[U_n \geq \left(1 - \frac{C_2}{\sqrt{\log n}}\right) N_n\right] \geq 1 - e^{-C_1 N_n/\sqrt{\log n}}.$$

This claim follows directly from a comparison with Bernoulli random variables : setting $I_n^{(i)} = 1$ if the spine decomposition of $X_n^{(i)}$ is unique, and $I_n^{(i)} = 0$ otherwise, the lower bound in (D.26) shows that each $I_n^{(i)}$ is stochastically bounded from below by a Bernoulli random variable with mean $1-C_7/\sqrt{\log n}$. Hence, U_n is stochastically bounded from below by the sum U_n' of N_n i.i.d. Bernoulli r.v., and a standard use of Markov's inequality gives us that if we choose $C_2 > 0$ large enough,

$$\mathbb{P}\left[U_n' < \left(1 - \frac{C_2}{\sqrt{\log n}}\right) N_n\right] \leq e^{-C_1 N_n/\sqrt{\log n}}$$

for a constant $C_2 > 0$ which can be computed explicitly in terms of C_2 and C_7. Therefore,

$$\mathbb{P}_n^H\left[U_n \geq \left(1 - \frac{C_2}{\sqrt{\log n}}\right) N_n\right] \geq \mathbb{P}_n^H\left[U_n' \geq \left(1 - \frac{C_2}{\sqrt{\log n}}\right) N_n\right] \geq 1 - e^{-C_1 N_n/\sqrt{\log n}}, \quad \text{(D.27)}$$

and the claim is proved.

Let us now focus on the fraction of trees for which the decomposition is unique. We want to show that with high probability, all of them will be pairwise distinct under height $n\sqrt{\log n}$ (and so pairwise distinct as a whole). To this end, let us use again T_n and its spine decomposition under the additional conditioning that all trees emanating from the spine under height $n\sqrt{\log n}$ are of height smaller than $n(\log n - \sqrt{\log n})$. We write $\tilde{\mathbb{P}}_n^H$ for the law of this tree. By construction, each brother of the i-th node of the spine ($0 \leq i \leq n\sqrt{\log n}-1$) has no offspring with probability

$$\begin{aligned} e_n &:= \mathbb{P}_n[T_n = \emptyset \,|\, H(T_n) \leq n(\log n - \sqrt{\log n})] \\ &= \frac{\mathbb{P}_n[T_n = \emptyset]}{\mathbb{P}_n[H(T_n) \leq n(\log n - \sqrt{\log n})]} \\ &= \frac{\ell_n(\{0\})}{\mathbb{P}_n[H(T_n) \leq n(\log n - \sqrt{\log n})]}. \end{aligned} \quad \text{(D.28)}$$

(Brother to the right or to the left does not matter here since the condition $H(T) < n(\log n - \sqrt{\log n})$ is stronger than $H(T) < n \log n - i - 1$ for our range of integers i.) Let us use (D.28) to obtain some bounds (away from 0 and 1), uniform in n and $i \leq n\sqrt{\log n} - 1$, for the probability that all of the $Y^n_{n \log n - i} - 1$ brothers of the i-th node of the spine have zero offspring. By the expression of $p_{n,2}$ and (D.25), the right-hand side of (D.28) is equivalent as $n \to \infty$ to

$$\frac{\ell_n(\{0\})}{1 - 2/(\sigma^2 n \log n)} \sim_{n\to\infty} \prod_{j=3}^{\infty} \left(1 - \frac{1}{q^j}\right) =: \mathbf{e} \in (0,1). \quad \text{(D.29)}$$

Thus, we have

$$\begin{aligned}\tilde{\mathbb{P}}_n^H[\text{no nephews at height } i] &\geq \mathbb{P}[Y^n_{n\log n-i} = q^3-q^2]\ e_n^{q^3-q^2-1}\\ &= \frac{1-\left(1-\frac{2}{\sigma^2(n\log n-i-1)}+o\left(\frac{1}{n\log n}\right)\right)^{q^3-q^2}}{2/(\sigma^2(n\log n-i))+o(1/n\log n)}\,\frac{\lambda(n-2)\lambda(n)}{\lambda(n-3)\lambda(1)\lambda(3)}q^{-3}e_n^{q^3-q^2-1}\\ &\sim \frac{q^3-q^2}{q^3}\frac{\mathbf{e}^{q^3-q^2-1}}{\lambda(1)\lambda(3)}\prod_{j=1}^{\infty}\left(1-\frac{1}{q^j}\right)\\ &= \mathbf{e}^{q^3-q^2-1}\prod_{j=4}^{\infty}\left(1-\frac{1}{q^j}\right)\in(0,1),\end{aligned}$$

where the last lines use (D.23), (D.25) and (D.29). Likewise,

$$\begin{aligned}\tilde{\mathbb{P}}_n^H[\text{at least one nephew at height } i] &\geq \mathbb{P}[Y^n_{n\log n-i} = q^3-q^2]\ (1-e_n^{q^3-q^2-1})\\ &\sim (1-\mathbf{e}^{q^3-q^2-1})\prod_{j=4}^{\infty}\left(1-\frac{1}{q^j}\right)\in(0,1).\end{aligned}$$

Hence, since these two probabilities belong to $(0,1)$ for all $n\geq 3$ and $i\leq n\sqrt{\log n}-1$, and belong to a smaller interval of $(0,1)$ bounded away from 0 and 1 whenever n is large enough, this provides the existence of $\kappa_l,\kappa_u\in(0,1)$ such that for every $n\geq 3$ and $i\in\{0,\dots,n\sqrt{\log n}-1\}$,

$$1-\kappa_l\leq \tilde{\mathbb{P}}_n^H[\text{no nephews at height } i]\leq \kappa_u. \tag{D.30}$$

Let x_1 be a topological tree of height at least $n\log n$, admissible for X_n (that is, all nodes have an offspring number of the form q^k-q^2) and such that its spine decomposition is unique under height $n\sqrt{\log n}$ (more precisely, we again ask that there is only one path from the root to height $n\sqrt{\log n}$ extending to a path reaching height $n\log n$, and that all trees emanating from brothers of the spine under height $n\sqrt{\log n}$ are of height smaller than $n(\log n-\sqrt{\log n})-1$). For every $i\in\{0,n\sqrt{\log n}-1\}$, let γ_i^n be the indicator function of the event that all brothers of the i-th node of the spine have no offspring. For every $n\geq 3$, $\{\gamma_i^n,\ 0\leq i\leq n\sqrt{\log n}-1\}$ form a family of independent r.v. and by (D.30), we have

$$\tilde{\mathbb{P}}_n^H[\gamma_i^n=1]\leq\kappa_u \qquad\text{and}\qquad \mathbb{P}_n^H[\gamma_i^n=0]\leq\kappa_l.$$

Comparing the absence or presence of nephews of the spine in X_n (or equivalently, in T_n) and in x_1, and setting $\kappa:=\max\{\kappa_l,\kappa_u\}<1$, we obtain

$$\tilde{\mathbb{P}}_n^H[X_n=x_1]\leq\kappa^{n\sqrt{\log n}}.$$

A direct consequence of the preceding inequality is that, if $x_1,\dots,x_k$ are k distinct trees satisfying the above properties, then

$$\tilde{\mathbb{P}}_n^H[X_n\notin\{x_1,\dots,x_k\}]=1-\sum_{j=1}^{k}\tilde{\mathbb{P}}_n^H[X_n=x_j]\geq 1-k\kappa^{n\sqrt{\log n}}. \tag{D.31}$$

Finally, let us set $\tilde{N}_n:=(1-\frac{C_2}{\sqrt{\log n}})N_n$ and suppose that $X_n^{(1)},\dots,X_n^{(\tilde{N}_n)}$ are $\tilde{N}_n$ independent Galton-Watson trees sampled according to the law $\tilde{\mathbb{P}}_n^H$. A straightforward

induction yields

$$\begin{aligned}\tilde{\mathbb{P}}_n^H\big[X_n^{(1)},\dots,X_n^{(\tilde N_n)}\text{ all distinct}\big] &= \prod_{j=2}^{\tilde N_n}\tilde{\mathbb{P}}_n^H\big[X_n^{(j)}\notin\{X_n^{(1)},\dots,X_n^{(j-1)}\}\,\big|\,X_n^{(1)},\dots,X_n^{(j-1)}\text{ all distinct}\big]\\ &\geq \prod_{j=2}^{\tilde N_n}\Big(1-(j-1)\kappa^{n\sqrt{\log n}}\Big)\\ &\geq \Big(1-\tilde N_n\kappa^{n\sqrt{\log n}}\Big)^{\tilde N_n},\end{aligned}\tag{D.32}$$

where the second inequality uses (D.31).

To conclude the proof of Theorem D.6, we combine all the preceding steps. Indeed, if we sample N_n independent Galton-Watson trees conditioned to have height at least $n\log n$, we know from (D.27) that with probability at least $1-e^{-C_1N_n/\sqrt{\log n}}$, at least $\tilde N_n$ of them will have a unique spine decomposition. Focusing on the $\tilde N_n$ first such trees, (D.32) show that with probability at least $\big(1-\tilde N_n\kappa^{n\sqrt{\log n}}\big)^{\tilde N_n}$ all of them will be pairwise distinct. This yields the desired result. □

Proof of Lemma D.8. Let us use the Markov inequality as follows : if $C_3>0$, we have for each $n\geq 3$

$$\begin{aligned}&\mathbb{P}\left[\sum_{i=0}^{n\sqrt{\log n}-1}Y^n_{n\log n-i}>C_3n\sqrt{\log n}\right]\\ &=\mathbb{P}\Big[\sum_{i=0}^{n\sqrt{\log n}-1}\big(Y^n_{n\log n-i}-\mathbb{E}[Y^n_{n\log n-i}]\big)>C_3n\sqrt{\log n}-\sum_{i=0}^{n\sqrt{\log n}-1}\mathbb{E}[Y^n_{n\log n-i}]\Big]\\ &\leq\frac{\mathbb{E}\Big[\Big(\sum_{i=0}^{n\sqrt{\log n}-1}\big(Y^n_{n\log n-i}-\mathbb{E}[Y^n_{n\log n-i}]\big)\Big)^2\Big]}{\Big(C_3n\sqrt{\log n}-\sum_{i=0}^{n\sqrt{\log n}-1}\mathbb{E}[Y^n_{n\log n-i}]\Big)^2}.\end{aligned}\tag{D.33}$$

Let us show that the numerator in the right-hand side of (D.33) is of order $n\sqrt{\log n}$, while the denominator is of order $n^2\log n$ whenever $C_3>0$ is large enough.

These two points rely on appropriate bounds on the first two moments of all $Y^n_{n\log n-i}$'s. Indeed, recall from (D.23) that for every $k\in\{3,\dots,n\}$,

$$\mathbb{P}\big[Y^n_{n\log n-i}=q^k-q^2\big]=\frac{1-\mathbb{P}_n\big[H(T_n)<n\log n-i-1\big]^{q^k-q^2}}{\mathbb{P}_n\big[H(T_n)\geq n\log n-i\big]}\frac{\lambda(n-2)\lambda(n)}{\lambda(n-k)\lambda(k-2)\lambda(k)}q^{-k(k-2)},$$

and these are the only possible values for $Y^n_{n\log n-i}$. Because $1-e^{-x}\leq x$ for all $x\geq 0$, we can write for every $i\leq n\sqrt{\log n}-1$:

$$\begin{aligned}1-\mathbb{P}_n\big[H(T_n)<n\log n-i-1\big]^{q^k-q^2}&\leq-(q^k-q^2)\log\big(\mathbb{P}_n\big[H(T_n)<n\log n-i-1\big]\big)\\ &\leq-(q^k-q^2)\log\big(\mathbb{P}_n\big[H(T_n)<n\log n-n\sqrt{\log n}-2\big]\big).\end{aligned}$$

We thus have for every such integer i

$$\frac{1-\mathbb{P}_n\big[H(T_n)<n\log n-i-1\big]^{q^k-q^2}}{\mathbb{P}_n\big[H(T_n)\geq n\log n-i\big]}\leq-(q^k-q^2)\frac{\log\big(\mathbb{P}_n\big[H(T_n)<n\log n-n\sqrt{\log n}-2\big]\big)}{\mathbb{P}_n\big[H(T_n)\geq n\log n\big]}.$$

Moreover,

$$\frac{\lambda(n-2)\lambda(n)}{\lambda(k-2)\lambda(k)\lambda(n-k)} = \frac{\prod_{j=k-1}^{n-2}(1-q^{-j})\prod_{j'=k+1}^{n}(1-q^{-j'})}{\prod_{j=1}^{n-k}(1-q^{-j})} \leq \Big(\prod_{j=1}^{\infty}\Big(1-\frac{1}{q^j}\Big)\Big)^{-1} =: C_q.$$

Combining the above, we arrive at

$$\mathbb{P}\big[Y^n_{n\log n-i} = q^k - q^2\big] \leq \frac{-\log\big(\mathbb{P}_n\big[H(T_n) < n\log n - n\sqrt{\log n} - 2\big]\big)}{\mathbb{P}_n\big[H(T_n) \geq n\log n\big]} C_q\, q^{-k(k-2)}(q^k - q^2)$$

for every $n \geq 3$ and $k \in \{2,\dots,n\}$. This yields

$$\mathbb{E}\big[Y^n_{n\log n-i}\big] \leq \frac{-\log\big(\mathbb{P}_n\big[H(T_n) < n\log n - n\sqrt{\log n} - 2\big]\big)}{\mathbb{P}_n\big[H(T_n) \geq n\log n\big]} C_q \sum_{k=3}^{n} q^{-k(k-2)}(q^k - q^2)^2 \qquad \text{and}$$

$$\mathbb{E}\big[(Y^n_{n\log n-i})^2\big] \leq \frac{-\log\big(\mathbb{P}_n\big[H(T_n) < n\log n - n\sqrt{\log n} - 2\big]\big)}{\mathbb{P}_n\big[H(T_n) \geq n\log n\big]} C_q \sum_{k=3}^{n} q^{-k(k-2)}(q^k - q^2)^3.$$

Now, by (D.25) we have

$$\lim_{n\to\infty} \frac{-\log\big(\mathbb{P}_n\big[H(T_n) < n\log n - n\sqrt{\log n} - 2\big]\big)}{\mathbb{P}_n\big[H(T_n) \geq n\log n\big]} = 1,$$

and furthermore,

$$\sum_{k=3}^{\infty} q^{-k(k-2)}(q^k - q^2)^2 =: m_1 < \infty \qquad \text{and} \qquad \sum_{k=3}^{\infty} q^{-k(k-2)}(q^k - q^2)^3 =: m_2 < \infty.$$

As a consequence, there exists $C > 0$ such that for every $n \geq 3$, we have

$$\sum_{i=0}^{n\sqrt{\log n}-1} \mathbb{E}\big[Y^n_{n\log n-i}\big] \leq C m_1 n\sqrt{\log n},$$

and (using the independence of all Y^n_m's)

$$\begin{aligned}\mathbb{E}\Big[\Big(\sum_{i=0}^{n\sqrt{\log n}-1}\big(Y^n_{n\log n-i} - \mathbb{E}\big[Y^n_{n\log n-i}\big]\big)\Big)^2\Big] &= \sum_{i=0}^{n\sqrt{\log n}-1} \mathrm{Var}\big(Y^n_{n\log n-i}\big)\\ &\leq C' n\sqrt{\log n},\end{aligned}$$

for a constant $C' > 0$ depending on m_1 and m_2. Choosing $C_3 > C m_1$ and coming back to (D.33), we obtain the existence of $C_4 > 0$ such that for every $n \geq 3$,

$$\mathbb{P}\Big[\sum_{i=0}^{n\sqrt{\log n}-1} Y^n_{n\log n-i} \geq C_3 n\sqrt{\log n}\Big] \leq C_4 \frac{n\sqrt{\log n}}{n^2\log n} = \frac{C_4}{n\sqrt{\log n}}.$$

This completes the proof of Lemma D.8. □

Attaques statistiques

Annexe E :

An Improved LPN Algorithm, SCN 2006

Éric Levieil et Pierre-Alain Fouque

Cet article présente une amélioration de l'algorithme de Blum, Kalai et Wasserman pour résoudre le problème Learning Parity with Noise (LPN). Nous avons remarqué que la dernière étape de l'attaque pouvait être accélérée en utilisant un calcul de transformée de Fourier. Cette amélioration permet de réduire la constante c dans la complexité sous-exponentielle, $2^{cn/\log n}$, de cet algorithme. Ceci nous a permis de montrer que certains choix de paramètres pour des systèmes basés sur ce problème n'était pas sûr.

Annexe F :

Second Preimage Attacks on Dithered Hash Functions, EUROCRYPT 2008

Elena Andreeva, Charles Bouillaguet, Pierre-Alain Fouque, Jonathan J. Hoch, John Kelsey, Adi Shamir et Sébastien Zimmer

Cet article présente une attaque sur une contremesure proposée par Rivest pour éviter une attaque en seconde préimage sur les fonctions de hachage utilisant le mode de construction de Merkle et Damgård. Nous avons développé une nouvelle attaque contre ce mode qui permet de mieux prendre en compte les modifications proposées par Rivest. Notre attaque utilise sur une structure de donnée, appelée diamant par Kelsey et Kohno, particulièrement bien adaptée aux attaques sur les fonctions de hachage.

Annexe G :

Full Key-Recovery Attacks on HMAC/NMAC-MD4 and NMAC-MD5, CRYPTO 2007

Pierre-Alain Fouque, Gaëtan Leurent et Phong Q. Nguyen

Cet article présente une application des attaques en collision de Wang sur les fonctions de hachage aux schémas d'authentification de message utilisant de telles fonctions. Nous avons montré que les chemins différentiels utiles pour trouver des collisions dans les fonctions de hachage peuvent aussi être utile pour obtenir de l'information sur les clés secrètes utilisées dans ce schéma. Notre attaque permet de retrouver toute la clé de HMAC-MD4 avec une complexité en 2^{95} en temps et 2^{88} en donnée et dans le modèle à clé liée, de NMAC-MD5 avec une complexité en 2^{51} en temps et 2^{100} en donnée.

Annexe E

An Improved LPN Algorithm

SCN 2006

[205] avec Éric Levieil

Abstract : *HB$^+$ is a shared-key authentication protocol, proposed by Juels and Weis at Crypto 2005, using prior work of Hopper and Blum. Its very low computational cost makes it attractive for low-cost devices such as radio-frequency identification (RFID) tags. Juels and Weis gave a security proof, relying on the hardness of the "learning parity with noise" (LPN) problem. Here, we improve the previous best known algorithm proposed by Blum, Kalai, and Wasserman for solving the LPN problem. This new algorithm yields an attack for HB$^+$ in the detection-based model with work factor 2^{52}.*

E.1 Introduction

Providing lightweight and secure cryptographic protocols for radio-frequency identification (RFID) tags is an area under quick development. The HB protocol family is one of the most promising in this field and uses very few operations and gates on the chip. The original protocol has been proposed by Hopper and Blum [154].

All protocols in the HB family rely on the computational hardness of the LPN problem.

The LPN Problem.

In machine learning theory, this problem is described in the *uniform distribution model* where the algorithm only has access to a source of random samples. The LPN problem is the following :

Définition E.1
(LPN Problem)
Let $\langle \cdot | \cdot \rangle$ denote the binary inner product. Let $\mathbf{s}$ be a random k-bit vector, let $\varepsilon \in]0, 1/2[$ be a constant noise parameter, let Ber_ε be the Bernoulli distribution with parameter ε (so if $\nu \leftarrow Ber_\varepsilon$ then $Pr[\nu = 1] = \varepsilon$ and $Pr[\nu = 0] = 1 - \varepsilon$), and let $A_{\mathbf{s},\varepsilon}$ be the distribution defined as

$$\left\{ \mathbf{a} \leftarrow \{0,1\}^k ; \nu \leftarrow Ber_\varepsilon : (\mathbf{a}, \langle \mathbf{s} | \mathbf{a} \rangle \oplus \nu) \right\}$$

Let $A_{\mathbf{s},\varepsilon}$ denote an oracle which outputs independent samples according to this distribution. Algorithm M is said to (q, t, m, θ)-solve the $LPN_{k,\varepsilon}$ problem if

$$Pr[\mathbf{s} \leftarrow \{0,1\}^k : M^{A_{\mathbf{s},\varepsilon}}(1^k) = \mathbf{s}] \geq \theta$$

and furthermore M runs in time at most t, memory at most m, and makes at most q queries to its oracle. ⧫

This is the definition of Regev [254], and Katz *et al.* [174]. An alternative (and equivalent) definition can be found for example in [172].

In the following, we define δ as $\delta = 1 - 2\varepsilon$. This notation will be better to analyze the complexity of the algorithms. For the classical parameters $\varepsilon = 1/4$ and $1/8$, δ is equal to $1/2$ and $3/4$.

The LPN problem is an average-case version of the following problem : given a set of equations over GF(2), find a vector s that maximally satisfies the equations. The latter problem has been first studied as the decoding of a random linear code and has been proved to be NP-hard by Berlekamp *et al.* in [26]. It has also be shown to be hard to approximate even within a factor of two by Hastad in [149] : finding $\mathbf{s}$ that satisfies more than half of the optimum number of equations is hard. In the LPN problem, the instances (set of equations and values) maybe do not represent the worst case of the problem, but studies of the average-case hardness of this problem have been proposed in [154, 176, 40, 41, 254].

The HB Protocol.

The Reader and the Tag share public values k, ε, u and r, and a k-bit secret value $\mathbf{s}$. To be authenticated by a Reader, the Tag and the Reader repeat the following round many times :

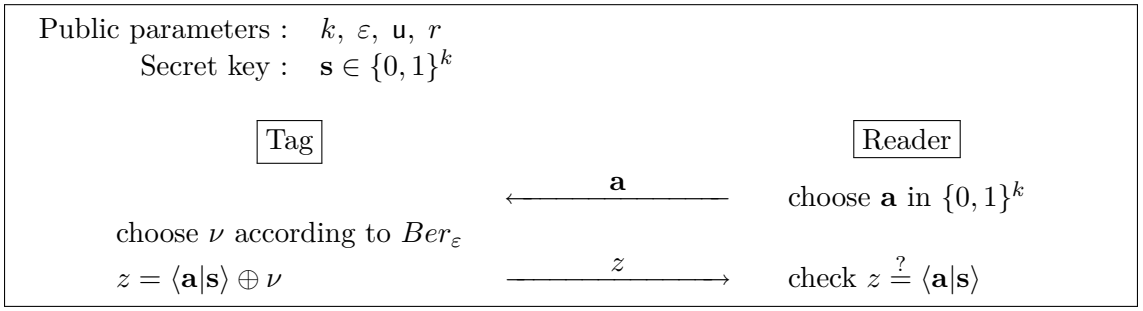

FIGURE E.1: Round identification of HB scheme

The round is repeated r times so that the Reader has good confidence in the answers of the Tag. To this end, the protocol has a parameter u so that if the number of errors is less than $r \cdot \mathsf{u}$, then the authentication is successful. Typical values of ε is $1/4$ or $1/8$. This value cannot be chosen too close to $1/2$, otherwise the probability of rejecting an honest Tag increases too much. If it is too close to 0, then you can find k independent equations without errors easily.

A completeness error occurs when an honest Tag is rejected. We want the probability of a completeness error, P_c to be less than 2^{-40}.

A soundness error occurs when a Tag making random answers succeeds in authenticating itself. We want the probability of a soundness error, P_s to be less than 2^{-80}.

ε	0.01	0.05	0.125	0.25	0.4	0.49
u	0.112	0.181	0.256	0.348	0.442	0.495
r	159	249	441	1164	7622	554360

FIGURE E.2: Values for $r(\varepsilon)$

Given ε, u, and r, we can compute the value of P_c and P_s. Let

$$g(x,y) = \left(\frac{x}{y}\right)^x \left(\frac{1-x}{1-y}\right)^{1-x}.$$

The probabilities P_c and P_s can be expressed as sums of the tail of a binomial distribution, and using Stirling's formula, we obtain :

$$P_c \sim g(\mathsf{u},\varepsilon)^{-r} \text{ and } P_s \sim g(\mathsf{u},1/2)^{-r}.$$

For each ε, we compute the values of u and r such that r is as small as possible, and the above conditions on P_c and P_s are true.

We gather the result in the following table :

In this protocol, secure only against passive attackers, an adversary gets pairs of the form $(\mathbf{a}, \langle \mathbf{a}|\mathbf{s}\rangle \oplus \nu)$ and must compute $\mathbf{s}$.

There is a simple active attack against HB. Indeed, if an adversary can change the challenge, it can send several times the same value a. Since the answer is incorrect with probability $\varepsilon \ll 1/2$, majority votes enables to recover $\langle \mathbf{a}|\mathbf{s}\rangle$. Then, k scalar products with independent $\mathbf{a}$, allow to entirely recover $\mathbf{s}$.

The HB$^+$ Protocol.

Consequently, Juels and Weis in [172] have proposed HB$^+$, a protocol robust against active attacks in the detection-based model. The idea is to use a blinding factor. The Tag and the Reader now share two k-bit secret values $\mathbf{s}_1$ and $\mathbf{s}_2$. The protocol round is the following :

Public parameters : $k,\ \varepsilon,\ \mathsf{u},\ r$
Secret key : $\mathbf{s}_1, \mathbf{s}_2 \in \{0,1\}^k$

Tag		Reader
choose $\mathbf{b}$ in $\{0,1\}^k$	$\xrightarrow{\mathbf{b}}$	
	$\xleftarrow{\mathbf{a}}$	choose $\mathbf{a}$ in $\{0,1\}^k$
choose ν according to Ber_ε		
$z = \langle \mathbf{a}|\mathbf{s}_1\rangle \oplus \langle \mathbf{b}|\mathbf{s}_2\rangle \oplus \nu$	$\xrightarrow{z}$	check $z \stackrel{?}{=} \langle \mathbf{a}|\mathbf{s}_1\rangle \oplus \langle \mathbf{b}|\mathbf{s}_2\rangle$

FIGURE E.3: Round identification of HB$^+$ scheme

The security of this protocol relies on the LPN problem for $\mathbf{s}_2$. Indeed, an active attacker can interact with the Tag in the first stage of his attack and then tries to impersonate the Tag against a Reader. The attacker can choose $\mathbf{a} = 0^k$, obtaining values of $\langle \mathbf{b}|\mathbf{s}_2\rangle \oplus \nu$. If he can solve a $LPN_{k,\varepsilon}$ problem, he can recover $\mathbf{s}_2$. Then, once $\mathbf{s}_2$ is recovered, he must

recover $\mathbf{s}_1$. This can be easily done since the attacker is now faced to a HB protocol that can be defeated by choosing the same $\mathbf{a}$ many times to known $\langle \mathbf{a}|\mathbf{s}_1 \rangle$ with high confidence.

Remarque E.1. The length of $\mathbf{s}_1$ is used in HB$^+$ proofs only to guarantee that the attack that consists to guess $\mathbf{s}_1$ is not efficient. But $|\mathbf{s}_1| = 80$ is sufficient to guarantee 80 bits security. ∗

Related Works.

Gilbert, Robshaw, and Sibert [138] found a man-in-the-middle attack against HB$^+$ when the adversary can interact with the Reader and the Tag during the same round.

The security proof of HB$^+$ of Juels and Weis has also been simplified and improved by Katz and Sun Shin [174], using a recent result of Regev [254].

Our work.

The best algorithm to solve the LPN problem had been proposed by Blum, Kalai and Wasserman in [41], hereafter denoted by the BKW algorithm. The parameter security k of the HB protocols has therefore been estimated using the complexity of this algorithm. But, Blum, *et al.* only give a high-level description of the BKW algorithm and estimate the overall subexponential complexity of order $2^{O(k/\log k)}$. Juels and Weis in [172] propose practical parameters by giving an effective estimate of the query and time complexity.

However, they have not seen that the BKW algorithm could be improved. In this paper, we present in detail the BKW algorithm, analyze it precisely and give its complexity. Then, we propose an improvement for the final stage of the algorithm. Instead of throwing away almost all equations, we manage to use every one. Therefore, we need much less queries. We also use a Walsh-Hadamard transform to speed up this phase. Then, we give an heuristic improvement using Wagner's method to solve the Generalized Birthday Paradox of [285]. Finally, we compare the performance of the BKW and our algorithm. Our algorithm yields an attack in 2^{52} for the actual key-length of HB$^+$ as proposed by Juels and Weis in Crypto '05, instead of the conjectured 2^{80}.

The next section is a full analysis of BKW. In the third one, we describe and prove an improved algoritm (LF1). In the fourth, we propose an heuristic algorithm (LF2) that is more efficient in practice than LF1, as will be shown in the last section, that is focused on implementation techniques and complexity results.

E.2 The BKW Algorithm

The aim of this section is to describe the algorithm and the ideas behind. We also make a detailed and precise analysis of the success probability, using explicit Chernoff's bounds that are recalled in appendix E.7. This part is not contained in the previous papers.

E.2.1 Description

In the following, we denote by a and b two different parameters from $\mathbf{a}$ and $\mathbf{b}$. We use those very close notations since the first come from Blum *et al.* in [41] and the second come from [172, 174].

To solve the learning parity with noise problem Blum, Kalai and Wasserman in [41] use the following idea : by picking carefully a few well-chosen vectors in a quite large set of samples and computing the xor of these vectors, we can find basis vectors, *i.e.* e_j where the

jth bit is a one and all other coordinates are null. First, we have to choose a parameter a. Typical values run from 4 to 6. The main point is that we are able to find $2^a = O(k/\log k)$ vectors such that

$$\mathbf{a}_{i_1} \oplus \cdots \oplus \mathbf{a}_{i_{2^a}} = e_j. \tag{E.1}$$

Then, since the number (2^a) of vectors is small, the bias of the equations obtained is not too small. Consequently, if we have enough independent combinations of vectors equals to e_j, then a majority vote enables us to recover the correct value of $\mathbf{s}_j$ since $\langle \mathbf{s} | \mathbf{a}_{i_1} \oplus \cdots \oplus \mathbf{a}_{i_{2^a}} \rangle = \langle \mathbf{s} | e_j \rangle = \mathbf{s}_j$.

We set b to be $\lceil \frac{k}{a} \rceil$. From now on, we will assume for simplicity that $k = a \cdot b$. The algorithm has to search enough such independent combinations of the $\mathbf{a}_i$'s. To this end, it splits the k bits of $\mathbf{a}_i$ into a blocks of b bits. Then, according to the last b bits, the algorithm computes 2^b equivalent classes and classifies the $\mathbf{a}_i$ according to these bits. In each class, it chooses a vector at random, performs the xor with all other vectors of the same class and finally throws away this vector. Therefore, at the end of this step, in each equivalent class, the last b bits are zeroes. This procedure is called recursively beginning at the last block until the second block. Then, we keep only the equations that are of the form $\langle \mathbf{s} | e_j \rangle = \nu$. If there are enough of such equations, the majority vote says something meaningful about the value of $\mathbf{s}_j$ with high probability. By applying this algorithm for different j, we can recover all the bits of $\mathbf{s}$.

E.2.2 Analysis

Now, we will analyse this algorithm. To this end, we present two lemmas that will be helpful. The first lemma analyses the bias at the end of the recursion steps, while the second lemma estimates the number of elements and is useful to show an invariant of the algorithm.

Lemme E.1. *If $(\mathbf{a}_1, \nu_1), \ldots, (\mathbf{a}_n, \nu_n)$ are the result of n queries to $A_{\mathbf{s},\varepsilon}$, then the probability that :*

$$\langle \mathbf{a}_{i_1} \oplus \cdots \oplus \mathbf{a}_{i_n} | s \rangle = \nu_{i_1} \oplus \cdots \oplus \nu_{i_n}$$

is equal to $\frac{1+\delta^n}{2}$. ▽

This lemma is equivalent to lemma 3 of [41].

Démonstration. For $n = 1$, the lemma is trivially true. By induction, and using the $\mathbf{a}_i$'s independence, we have :

$$\begin{aligned} &Pr[\langle \mathbf{a}_{i_1} \oplus \cdots \oplus \mathbf{a}_{i_n} | \mathbf{s} \rangle = \nu_{i_1} \oplus \cdots \oplus \nu_{i_n}] = \\ &Pr[\langle \mathbf{a}_{i_1} \oplus \cdots \oplus \mathbf{a}_{i_{n-1}} | \mathbf{s} \rangle = \nu_{i_1} \oplus \cdots \oplus \nu_{i_{n-1}}] Pr[\langle \mathbf{a}_{i_n} | \mathbf{s} \rangle = \nu_{i_n}] \\ &+ Pr[\langle \mathbf{a}_{i_1} \oplus \cdots \oplus \mathbf{a}_{i_{n-1}} | \mathbf{s} \rangle \neq \nu_{i_1} \oplus \cdots \oplus \nu_{i_{n-1}}] Pr[\langle \mathbf{a}_{i_n} | \mathbf{s} \rangle \neq \nu_{i_n}] \\ &= \frac{1}{2}(1 + \delta^{n-1}) \frac{1}{2}(1 + \delta) + \frac{1}{2}(1 - \delta^{n-1}) \frac{1}{2}(1 - \delta) = \frac{1}{2}(1 + \delta^n). \end{aligned}$$

■

The following definition and lemma are equivalent to definition 2 and lemma 4 in [41].

Définition E.2
Let $A_{\mathbf{s},\delta,i}$ be the distribution defined as

$$\left\{ \mathbf{a} \leftarrow \{0,1\}^{(a-i)b} \times \{0\}^{ib}; \nu \leftarrow Ber_{(1+\delta)/2} : (\mathbf{a}, \langle \mathbf{s} | \mathbf{a} \rangle \oplus \nu) \right\}$$

⧫

Also, let $A_{\mathbf{s},\delta,i}$ denote an oracle which outputs independent samples according to this distribution. We define an $(\mathbf{s},\delta,i)$-set of size n as the result of n queries to oracle $A_{\mathbf{s},\delta,i}$.

Lemme E.2. *Assume we are given an $(\mathbf{s},\delta,i)$-set of size n. We can in time $O(n)$ construct an $(\mathbf{s},\delta^2,i+1)$-set of size $n-2^b$.* ▽

Démonstration. Let us call $(\mathbf{a}_1,\nu_1),\dots(\mathbf{a}_n,\nu_n)$ the elements of the $(\mathbf{s},\delta,i)$-set. Vectors $\mathbf{a}_j$ have their last ib coordinates equal to 0. We partition them with regard to their value on the precedent b coordinates, obtaining a partition with at most 2^b classes. In each class, we pick a vector at random and add it (modulo 2) to all the others vectors in that class, and then discard it. Compiling the results for each class, and using lemma E.1, we obtain a $(\mathbf{s},\delta^2,i+1)$-set of size (at least) $n-2^b$. ■

Consequently, according to lemma E.1, at the end of the algorithm, the bias of the equation I.1 is $\delta^{-2^{a-1}}$ where $\delta=(1-2\varepsilon)$.

The next lemma give the number of combinations of equations that must xored to the vector e_j in order to have a high probability of success.

Lemme E.3. *Let $A_{\mathbf{s}_i,\delta,a}$ be the distribution defined as*

$$\left\{\nu \leftarrow Ber_{(1+\delta^{2^{a-1}})/2} : \mathbf{s}_i \oplus \nu)\right\}$$

Also, let $A_{\mathbf{s}_i,\delta,a}$ denote an oracle which outputs independent samples according to this distribution.

Then it is possible to guess the value of $\mathbf{s}_i$ with $c\delta^{-2^a}$ calls to the oracle with error probability bound by $2e^{-c/20}$. ▽

Démonstration. We define that a sample $\mathbf{a}_i,\nu$ is *compatible* with the i-th bit of $\mathbf{s}$ if $\mathbf{s}_i\cdot\mathbf{a}_i=\nu$.

The idea for guessing the ith bit is to compute for $\mathbf{s}_i=0$ and $\mathbf{s}_i=1$, the number of compatible samples and predict that $\mathbf{s}_i=b$ according to the majority number of compatible samples.

Therefore, in order to upper bound the probability of failure of the BKW algorithm, we have to upper bound the following probability where $\mathbf{x}_i=1-\mathbf{s}_i$:
$\Pr[\mathbf{x}_i$ has more compatible samples than $\mathbf{s}_i]$.

To this end, we upper bound the previous probability by the sum of two more easily computable probabilities.

$$\mathsf{pr}_1=\Pr[\mathbf{x}_i \text{ is compatible with at least } \frac{1+\alpha\delta^{2^{a-1}}}{2}\cdot N \text{ samples}]$$
$$\mathsf{pr}_2=\Pr[\mathbf{s}_i \text{ is compatible with at most } \frac{1+\alpha\delta^{2^{a-1}}}{2}\cdot N \text{ samples}]$$

If $\mathbf{s}_i$ is correct, then it is compatible with $\mathbf{a}_i,\nu$ with probability $\frac{1+\delta^{2^{a-1}}}{2}$ and otherwise with probability $1/2$. We will justify the last assertion later. Let us denote by N the number of equations $c\delta^{-2^a}$. The random variable X_j for $j=1$ to N, is equal to 1 if $\mathbf{s}_i$ is compatible with the jth sample, and 0 otherwise.

Bounding pr_2.

The expectation of X_j, $\mathbf{E}[X_j] = \Pr[X_j = 1] = \frac{1+\delta^{2^{a-1}}}{2}$ We sum these random variables and denote by X their sum, $X = \sum_{j=1}^{N} X_j$, and so $\mathbf{E}[X] = N \cdot \mathbf{E}[X_j] = N \cdot \frac{1+\delta^{2^{a-1}}}{2}$.

pr_2 is equal to $\Pr[X \leq (1+\alpha\delta^{2^{a-1}})(N/2)]$ which can be bounded using Chernoff bounds (cf. appendix E.7). To this end, we have $(1-\Delta)\mathbf{E}[X] = (1+\alpha\cdot\delta^{2^{a-1}})\cdot(N/2)$.

To determine Δ, we divide the right-hand side by $\mathbf{E}[X]$, and we get

$$1-\Delta = \frac{1+\alpha\cdot\delta^{2^{a-1}}}{1+\delta^{2^{a-1}}} \approx 1-(1-\alpha)\cdot\delta^{2^{a-1}}$$

and so $\Delta = (1-\alpha)\cdot\delta^{2^{a-1}}$.

$$\begin{aligned}\mathsf{pr}_2 &\leq e^{-(c\delta^{-2^a}/4)\cdot(1+\delta^{2^{a-1}})\cdot(1-\alpha)^2\delta^{2^{a-1}}\cdot 2} \leq e^{-(c/4)(1-\alpha)^2(1+\delta^{2^{a-1}})}\\ &\leq e^{-(c/4)(1-\alpha)^2}\end{aligned}$$

Bounding pr_1.

In order to upper bound pr_1, we use the fact that for a bad guess, the expectation $\mathbf{E}[X]$ is equal to $\frac{N}{2}$, and the theorem E.6 in appendix E.7. We have

$$\mathsf{pr}_1 \leq \Pr[X > (1+\Delta)\mu] \leq e^{-N\Delta^2/(3\cdot 2)}$$

Here, $\Delta = \alpha\cdot\delta^{2^{a-1}}$ and as $N = c\delta^{-2^a}$, then $N\Delta^2 = c\alpha^2$ and

$$\mathsf{pr}_1 \leq e^{-c\alpha^2/6}.$$

The probability $\Pr[\mathbf{x}_i\cdot\mathbf{a}_i = \nu | \mathbf{x}_i = 1-\mathbf{s}_i]$ is equal to $1/2$.

It remains to justify that when $\mathbf{s}_i$ is not correct, then $\mathbf{s}_i\cdot\mathbf{a}_i = \nu$ with probability $1/2$. Let $(\mathbf{a},\nu) \leftarrow A_{\mathbf{s},\varepsilon}$ and $\mathbf{x}_i = 1-\mathbf{s}_i$. We want to show that $\Pr[\mathbf{x}_i\cdot\mathbf{a}_i = \nu] = 1/2$. To this end, we split the event into two incompatible events :

$$\begin{aligned}\Pr[\mathbf{x}_i\cdot\mathbf{a}_i = \mathbf{s}_i\cdot\mathbf{a}_i]\Pr[\mathbf{s}_i\cdot\mathbf{a}_i = \nu] &+ \Pr[\mathbf{x}_i\cdot\mathbf{a}_i \neq \mathbf{s}_i\cdot\mathbf{a}_i]\Pr[\mathbf{s}_i\cdot\mathbf{a}_i \neq \nu]\\ &= \Pr[(\mathbf{x}_i\oplus\mathbf{s}_i)\cdot\mathbf{a}_i = 0]\Pr[\mathbf{s}_i\cdot\mathbf{a}_i = \nu]\\ &\quad + \Pr[(\mathbf{x}_i\oplus\mathbf{s}_i)\cdot\mathbf{a}_i \neq 0]\Pr[\mathbf{s}_i\cdot\mathbf{a}_i \neq \nu]\\ &= (1/2)\cdot(\Pr[\mathbf{s}_i\cdot\mathbf{a}_i = \nu] + \Pr[\mathbf{s}_i\cdot\mathbf{a}_i \neq \nu])\\ &= 1/2\end{aligned}$$

since the first equation comes from the fact that $\mathbf{a}_i$ and ν are independent and second equation as since $\mathbf{x}_i \neq \mathbf{s}_i$ and $\mathbf{a}_i$ is taken uniformly, the probability that $\mathbf{a}_i = 0$ is exactly $1/2$.

Choosing $\alpha = 3-\sqrt{6}$ finishes the proof. ∎

The main ingredient of the algorithm is that with a small number of vectors, a combination of such vectors yields a basis vector. If the number required is too high, then the bias of equation (I.1) is too small and the number of queries becomes very large.

We are now ready to prove the theorem that gives the complexity of the BKW algorithm.

Théorème E.4. *For $k = a \cdot b$, the BKW algorithm ($q = 20 \cdot \ln(4k) \cdot 2^b \cdot \delta^{-2^a}, t = O(kaq), m = kq, \theta = 1/2$)-solves the $LPN_{k,\varepsilon}$ problem .*

Démonstration. The original queries form a $(\mathbf{s}, \delta, 0)$-set of size q. Using lemma E.2 $(a-1)$ times, we obtain a $(\mathbf{s}, \delta^{2^{a-1}}, 0)$-set of size $q - (a-1)2^b$. Keeping only the equations with one non-zero coordinate, then using lemma E.3, we obtain one bit of $\mathbf{s}$ with error probability at most $1/(2k)$. Repeating this for different bits of $\mathbf{s}$, we find $\mathbf{s}$ with probability at least $1/2$. ∎

E.3 An Improved Algorithm : LF1

This algorithm is a variation of the BKW algorithm. In the BKW algorithm, the last step wastes a lot of time and queries. The idea is to deal in the last step with equations over b bits instead of one. Moreover, we will use the Walsh-Hadamard transform to quickly find the best possibility over b bits.

This algorithm does not use any heuristics. This section is devoted to prove the correctness and the performances of this algorithm. It needs lots of queries (less than BKW, though).

We now state our main theorem :

Théorème E.5. *For $k = a \cdot b$, there is an algorithm that*
$q = (8b + 200) \cdot \delta^{-2^a} + (a-1)2^b, t = O(kaq), m = kq + b2^b, \theta = 1/2$
solves the $LPN_{k,\varepsilon}$ problem.

Démonstration. For any b-bit vector $\mathbf{x}$, we say that a sample $\mathbf{a}_i, \nu$ is $\mathbf{x}$-compatible if $\langle \mathbf{a}_i | \mathbf{x} \rangle = \nu$. The q inital queries constitues a $(\mathbf{s}, \delta, 0)$-set of size q. By iterating lemma E.2, we obtain an $(\mathbf{s}, \delta^{2^{a-1}}, a)$-set S of size $q - (a-1) \cdot 2^b = N = (8b + 200)\delta^{-2^a}$.

We now try every possibility for the first b bits and choose the one that is compatible with the greatest number of examples. The naive time complexity is 2^{2b}, but using a fast Walsh-Hadamard transform reduces it to $b2^b$.

Using the same analysis as for the BKW algorithm, except that we choose $\alpha = 3/4$, we get that the probability of failure is less than

$$e^{-200/64} + 2^b e^{-(8b+200)9/96} \leq 1/(2a).$$

Repeating this a times allows us to recover all the bits of $\mathbf{s}$ with probability at least $1/2$. ∎

In this section, we have shown that we can lower the query complexity to $q = (a-1)2^b + (8b + 200)\delta^{-2^a}$. Time and memory complexity remains comparatively small.

E.4 A Heuristic Algorithm : Computing all sums : LF2

Following [285], instead of picking a vector in each class (cf. proof of E.2), we could compute the sum of any couple of class elements. Unfortunately, we lose the independence that is necessary to use Chernoff bounds. However, linear relations between equations are not numerous and our implementation confirms this phenomenon has no visible effect on the success of the algorithm.

This also allows us to overcome a lack of queries : if there are only $2^{b'}(b' > b/2)$ queries available, the first partitioning is made according to the last b_1 bits where $b_1 = 2b' - b - 1.5$. And for all the subsequent phases, we will have 2^b equations. Bounding the number of requests was an easy and effective defence against BKW and LF1, but it does not work against this new version.

E.5 Implementation

We want to do the partitioning, using only small additional memory and (almost) linear time. First, we divide our memory in $2^{b/2}$ packs of size $3.2^{b/2}$. We begin at the first equation in the first pack. Its last $b/2$ bits give the address of the pack where we send it. Here, it takes the place of an other equation, which is sent to the pack corresponding to its last $b/2$ bits, and so on. It could happen that a pack is full. In this case, the equation is lost. But few equations are lost in the process, and very few if the packs are a little bigger.

We now use an array of size $2^{b/2}$, each case being able to contain 10 equations. We put the equations of the first pack in this array, according to the values of bits $2^{b/2} + 1$ to 2^b(in reverse order). We could afford ten equations that have the same value (the average being 3).

Then we compute the xor between the first and the others for LF1, or the xor of all couples of equations in the same case for LF2, and put our new equations back to the pack.

We make an implementation in C, and make it run on a Pentium 4, with a CPU frequency of 3 GHz, and a little less than 1 GB of memory.

For $\varepsilon = 1/4$, using 1 GB of memory, our implementation breaks a LPN problem with $k = 99$ (we split the equations in four parts of sizes 24,24,27,24) with LF2, instead of a theoretic $k = 96$ with LF1. But we were able to break only a $k = 92$ with our implementation of LF1 because we need additional memory for pointers to equations. In both cases, the computing time was around 30 seconds.

E.5.1 Accurate complexity

The factor $8b + 200$ in the complexity is a rough upper bound. It can often be replaced by 25. On the other hand, reading and writing in a large memory (1 GB for example) could take tens of cycle per 32-bit int. If one uses a hard drive's memory, it will be a lot worse, even if one programs very carefully to make almost only sequential access to the drive.

E.5.2 Performances of our algorithm

First, we give a comparison between BKW and LF1.

For $\varepsilon = 1/4$, we have the following results :

The value given is the maximum value for k you could hope to break with the given memory. The needed time is roughly the time for sorting the memory a times.

Memory available	BKW	LF1
1 GB	39	96
2^{52} bytes	104	225
2^{80} bytes	180	426

The following tables contains a more exhaustive study of LF1. It should be read in the following way : It takes 2^{46} bytes of memory to solve a LPN problem with $k = 256$ and $\varepsilon = 1/8$.

$\varepsilon \backslash k$	64	128	256	512	768	1024
0.01	13	19	33	56	74	98
0.05	16	25	40	67	90	118
0.125	18	29	46	77	113	131
0.25	24	34	55	89	131	150
0.4	28	45	66	106	157	174
0.49	33	55	88	130	192	208

We suggest to take a safety margin, in order to be able to resist to small improvements like LF2. We have explored a variety of other improvements, but none of them gave substantial results.

We recommend to use $k = 512$ to achieve 80 bits security for $\varepsilon = 1/4$. Choosing the value of ε depends upon a compromise between the key size and the computing time for an authentication. Using tables E.2 and the above one should help. The couple $k = 768, \varepsilon = 0.05$ with $r = 249$ seems quite good.

E.6 Conclusion

In this article, we give a better algorithm to solve the LPN problem, thus breaking the HB+ protocol with suggested size parameters. However, with a moderate increase of the key length, our attack becomes infeasible. Our algorithm gives a more precise idea of the complexity of the LPN problem.

On the other side, remark E.1 allows to decrease the length of one of the secret parameters.

So, summing everything, we have shown that to achieve 80 bits security, HB^+ should be used with $|\mathbf{s}_1| = 80$ and $|\mathbf{s}_2| = 512$ instead of $|\mathbf{s}_1| = |\mathbf{s}_2| = 224$. The overall complexity of this protocol remains almost unchanged.

Proving algorithm LF2 is in our opinion quite difficult although feasible.

Acknowledgment. We would like to thank Louis Granboulan for various discussions and suggestions about this work.

E.7 Appendix Chernoff's Bounds

We need the following Chernoff's bounds that have been proved in [220].

Théorème E.6. *Let $X_1, \ldots, X_n$ be n independent Bernoulli trials such that $\Pr(X_i) = p$. Let $X = \sum_{i=1}^{n} X_i$ and $\mu = \mathbf{E}[X] = n \cdot p$. Then, the following Chernoff bounds hold :*

1. *for $0 < \Delta \leq 1$, $\Pr(X \geq (1 + \Delta)\mu) \leq e^{-\mu\Delta^2/3}$*
2. *for $0 < \Delta < 1$, $\Pr(X \leq (1 - \Delta)\mu) \leq e^{-\mu\Delta^2/2}$* ♢

Annexe F

Second Preimage Attacks on Dithered Hash Functions

EUROCRYPT 2008

[7] avec Elena Andreeva, Charles Bouillaguet, Jonathan J. Hoch, John Kelsey, Adi Shamir et Sébastien Zimmer

Abstract : *The goal of this paper is to analyze the security of dithered variants of the Merkle-Damgårdmode of operation that use a third input to indicate the position of a block in the message to be hashed. These modes of operation for hash functions have been proposed to avoid some structural weaknesses of the Merkle-Damgard paradigm, e.g. that second preimages can be constructed in much less than 2^n work, as pointed out by Kelsey and Schneier. Among the modes of operation that use such a third input are Rivest's dithered hashing and Biham and Dunkelman's* HAIFA*proposal.*
We propose several new second preimage attacks on the Merkle-Damgårdmode of operation, which can also attack Rivest's dithered hash with almost the same complexity. When applied to Shoup's UOWHF, these attacks can be shown to be optimal since their complexity matches Shoup's security bound.

F.1 Introduction

Hash functions have recently been the subject of numerous attacks which have highlighted many weaknesses (either on some specific hash functions or on the general Merkle-Damgårdmode of operation). Wang *et al.* [290, 291, 292, 293], Biham *et al.* [34], Klima [187] and Joux *et al.* [168] all show that differential attacks can be used to efficiently find collisions in specific hash functions based on the MD4 design such as MD5, RIPEMD, SHA-0 and SHA-1. This type of results is important for at least two reasons. First, collision resistance is a classical property that hash function should have. In addition, if collisions were easy to construct, perhaps it would become easier to mount stronger types of attacks which rely on the possibility to find collisions, such as Joux's [163] multicollision attack on combiners. Following this result, Kelsey and Schneier [179] extended a previous result of Dean [87], and showed that Joux's idea can be used to mount an efficient second preimage attack. All these attacks led the NIST to organize workshops and to solicit new hash function candidates.

After Kelsey and Schneier published their attack, several researchers proposed to tweak the Merkle-Damgårdparadigm, using a third input (called "dithering") to avoid attacks based on fixed points that can be iterated an arbitrary number of times. In this paper, we study the second preimage resistance of such dithered modes of operation, which is crucial for the security of signature schemes based on the "hash-and-sign" paradigm.

F.1.1 Related Work

Independently of the Ph.D thesis of Dean [87], Kelsey and Schneier [179] presented at EUROCRYPT 2005 a second preimage attack that works against all hash functions based on the Merkle-Damgårdconstruction. The complexity of their attack is $k \cdot 2^{n/2+1} + 2^{n-k+1}$ evaluations of the compression function. It relies on *expandable messages*, *i.e.* a family of messages of different lengths that hash to the same value. These messages are used to bypass the Merkle-Damgårdstrengthening. This in turn allows them to reapply the long message attack [215] which was supposedly foiled by the strengthening.

Biham and Dunkelman proposed HAIFA [232], which adds in each block the number of message bits which were hashed so far. The simplest way to implement HAIFAis to shorten each data block by 64 bits, and to concatenate the 64 bit counter instead. Rivest [255] introduced a clever way to decrease the number of bits used for this extra input to either 2 or 16, thus increasing the bandwidth available for actual data, by using a specific sequence of values to "dither" the actual inputs. The properties of this sequence were claimed by Rivest to be sufficient to avoid the Kelsey-Schneier attack.

Another variant of the long message attack is the "Nostradamus attack" of Kelsey and Kohno [178] which makes it possible to commit to a hash value h and then to find a message that hashes to h with any desired prefix. For their attack, they have introduced the "diamond" structure which is reminiscent of a binary tree. It is a 2^ℓ-multicollision where all the 2^ℓ colliding messages have a different initial value. Its construction cost is heavy, $2^{n/2+\ell/2+2}$ (where n is the size of the hash function output), but then it allows to connect any message to the collision tree with only $2^{n-\ell+1}$ calls to the hash function.

F.1.2 Our Results

In this paper, we propose several new generic second preimage attacks on various dithered versions of the Merkle-Damgårdparadigm in which the compression function can be modeled by a random oracle. Our technique mainly relies on the diamond structure from the herding attack of [178]. If the diamond is a 2^ℓ-multicollision, we obtain a second preimage of a message of size 2^k in time $2^{n/2+\ell/2+2} + 2^{n-\ell+1} + 2^{n-k+1}$. This expression is optimal for $\ell \approx n/3$ where the complexity becomes $2^{2n/3+2} + 2^{n-k+1}$. This is slightly more expensive than the $k \cdot 2^{n/2+1} + 2^{n-k+1}$ complexity of the Kelsey-Schneier attack (for SHA-1, the Kelsey-Schneier attack complexity is about 2^{106} work whereas ours is approximately 2^{109}). The main advantage of our attack is that it can be extended to Rivest's dithered proposal since the dithering sequence used in the third input has an undesirable property that can be exploited to make the attack efficient : the number of ℓ-letter subwords in it is exceptionally small, and thus one of these subwords must occur very frequently. Compared to the original attack, we lose a factor equal to the inverse of the frequency of the most probable word. The function mapping ℓ to the number of ℓ-letter words in the sequence is called the *factor complexity* of the sequence (see for example [4]). For the particular sequence chosen by Rivest, this complexity is only linear in the size ℓ of the words. For example, for Rivest's 16-bit sequence, the attack requires $2^{n/2+\ell/2+2} + (8\ell + 32768) \cdot 2^{n-\ell+1} + 2^{n-k+1}$ work, which for SHA-1 is approximately 2^{121}.

This is slightly worse than the attacks against the basic Merkle-Damgårdconstruction but it is still much smaller than the 2^{160} security which was expected for the dithered construction.

An extension of the basic attack can also be applied to a Universal One Way Hash Function designed by Shoup [270], which has some similarities with dithered hashing. Our technique yields the first published attack against this particular hash function. This additionally proves that Shoup's security bound is tight since there is asymptotically only a factor of $\mathcal{O}(\log k)$ between his bound and our attack's complexity.

The attacks presented above demonstrate that the properties of the dithering sequence influencing the security of dithered hash functions may not be well understood, and may include additional properties beyond its factor complexity and its repetition-freeness.

F.1.3 Organization of the Paper.

We describe our attack against the Merkle-Damgårdconstruction in section F.2. We introduce some terminology and describe the dithered Merkle-Damgårdconstruction in section F.3, and then we extend our attack to tackle dithered Merkle-Damgårdin section F.4. We apply it to Rivest's concrete proposal, as well as to some of the variations that he suggested. In section F.5, we apply the extended attack against Shoup's UOWHF construction.

F.2 A New Generic Second Preimage Attack

F.2.1 The Merkle-Damgårdconstruction

We first describe briefly the classical Merkle-Damgårdconstruction. An iterated hash function $H^F : \{0,1\}^* \to \{0,1\}^n$ is built by iterating a basic compression function $F : \{0,1\}^m \times \{0,1\}^n \to \{0,1\}^n$. The hash process works as follows :

- Pad and split a message M into r blocks $x_1, \ldots, x_r$ of m bits each.
- Set h_0 to the initialization value IV.
- For each message block i compute $h_i = F(h_{i-1}, x_i)$.
- Output $H^F(M) = h_r$.

The padding is usually done by appending a single '1' bit followed by as many '0' bits as needed to complete an m-bit block. Merkle [216] and Damgård [85] independently proved in 1989 that making the binary encoding of the message length part of the padding improves the security of the construction : with this so-called *strengthening*, the scheme is proven to be Collision-Resistance Preserving, in the sense that a collision in the hash function H^F would imply a collision in the compression function F. As a side effect, the strengthening defines an upper bound on the size of the messages that can be processed (because the encoding of the length has a fixed size). In most deployed hash functions, this limit is 2^{64} bits, or equivalently 2^{55} 512-bit blocks. In the sequel, we denote the maximal number of admissible blocks by 2^k.

F.2.2 Description of the Attack

We describe a new technique to build expandable messages. It relies heavily on the diamond structure introduced by Kelsey and Kohno [178] to find second preimages. An expandable message $\mathcal{M}$ is a family of messages with different number of blocks but with the same hash up to strengthening. We call any one of these messages an *instance* of $\mathcal{M}$. The *range* of $\mathcal{M}$ is the set of lengths of its instances.

A diamond of size ℓ is a multicollision that has the shape of a complete converging binary tree of height ℓ. It therefore has 2^ℓ leaves. Its nodes are labelled by chaining values over n bits, and its edges are labelled by message blocks over m bits, which map between the chaining values at the two ends of the edge by the compression function. Thus, from any one of the 2^ℓ leaves, there is a path labelled by ℓ message blocks that leads to the same target value h_T labelling the root of the tree.

Let M be a message of size 2^k. The main idea of the attack is that after a big collision tree of height ℓ is built during an (expensive) preprocessing stage, it costs less than 2^n work to "connect" the target chaining value h_T of the tree to one of the 2^k chaining values h_i obtained when hashing M. It also costs much less than 2^n work to connect an arbitrary prefix to one of the 2^ℓ leaves, if we do not care which one it is. The attack works in four steps :

1. Preprocessing step : compute a collision tree of height ℓ with an arbitrary target value h_T. Note that this has to be done only once, and can be reused when computing second preimages of multiple messages.
2. Connect the target h_T to some block in the message M. This can be done by generating random message blocks B, until $F(h_T, B) = h_{i_0}$ for some i_0, $\ell+1 \leq i_0 < |M|$. Let B_0 be a message block satisfying this condition.
3. Generate an arbitrary prefix P of size $i_0 - \ell - 1$ whose hash is one of the chaining values labelling a leaf. Let $h = H^F(P)$ be this value, and let T be the chain of ℓ blocks traversing the tree from h to h_T.
4. Form a message $M' = P||T||B_0||M_{i_0+1} \dots M_{2^r}$.

FIGURE F.1: Summary of the attack on classic Merkle-Damgård.

The message M' has the same hash as M, before strengthening, and has the same length, so we have obtained a full collision in the strengthened Merkle-Damgårdconstruction.

A collision tree of height ℓ can be constructed with time and space complexity $2^{\frac{n}{2}+\frac{\ell}{2}+2}$ (see [178] for details). The second step of the attack can be carried out with 2^{n-k+1} work, and the third one with $2^{n-\ell+1}$ work. The total time complexity of the attack is then :

$$2^{\frac{n}{2}+\frac{\ell}{2}+2} + 2^{n-k+1} + 2^{n-\ell+1}.$$

If n is much bigger than k (which is the case in all the members of the SHA family), then the first term of this sum becomes negligible compared to the other two ; we will use this approximation in the sequel. The complexity is minimal when $\ell = \frac{n}{3}$, and with this setting, the total cost of our attack is about $3 \cdot 2^{2n/3+2} + 2^{n-k+1}$.

It must be noted that if we choose $\ell \geq k$, then the connection in the third step of the attack can be realized for free. This step amount to finding a prefix of a given length, the hash of which is one of the values labelling the leaves of the collision tree. When there are more leaves than possible lengths for the prefix (that is, 2^k), then the hash value labelling the i-th leaf can be obtained by hashing a chosen message of length i blocks. This makes the attack slightly less expensive when n is very big.

F.2.3 Comparison With Kelsey and Schneier

The difference between the two techniques lies in the way expandable messages are built. Kelsey and Schneier build an expandable message in time $k \cdot 2^{n/2+1}$. Here, we build an expandable message in time $2^{n/2+\ell/2+2}+2^{n-\ell+1}$. On the original Merkle-Damgårdconstruction

their attack is more efficient than ours (on SHA-1, they can find a second preimage of a message of size 2^{55} with 2^{106} work, whereas we need 2^{109} calls to the compression function to obtain the same result). However, our technique to build expandable messages is more flexible, and in particular it can be adapted to work even when an additional dithering input is given, unlike the original Kelsey-Schneier technique. In addition, our technique gives the adversary more control on the second preimage, since she can typically choose about half of the message in an arbitrary way.

F.3 Dithered Hashing

The general idea of dithered hashing is to perturb the hashing process by using an additional input to the compression function, formed by the consecutive elements of a fixed *dithering* sequence. This gives the attacker less control over the input of the compression function, and makes the hash of a message block dependent on its position in the whole message. In particular, its goal is to prevent attacks based on expandable messages.

Since the dithering sequence $\vec{\mathbf{z}}$ has to be at least as long as the maximal number of blocks in any message that can be processed by the hash function, it is reasonable to consider infinite sequences as candidates for $\vec{\mathbf{z}}$. Let $\mathbf{a}$ be a finite alphabet, and let the dithering sequence $\vec{\mathbf{z}}$ be an eventually infinite word over $\mathbf{a}$. Let $\vec{\mathbf{z}}[i]$ denote the i-th element of $\vec{\mathbf{z}}$. The dithered Merkle-Damgårdconstruction is obtained by setting $h_i = F\left(x_i, h_{i-1}, \vec{\mathbf{z}}\left[i\right]\right)$ in the definition of the Merkle-Damgårdscheme.

F.3.1 Words and Sequences

Notations and Terminology.

Let ω be a word over the finite alphabet $\mathbf{a}$. The dot operator denotes concatenation. If ω can be written as $\omega = x.y.z$ (where x,y or z can be empty), we say that x is a *prefix* of ω and that y is a *factor* (or subword) of ω. A finite word ω is a *square* if it can be written as $\omega = x.x$, where x is not empty. A finite word ω is an *abelian square* if it can be written as $\omega = x.x'$ where x' is a permutation of x (*i.e.* a reordering of the letters of x). A word is said to be *square-free* (resp. *abelian square-free*) if none of its factors is a square (resp. an abelian square). Note that abelian square-free words are in particular square-free.

An Infinite Abelian Square-Free Sequence.

In 1992, Keränen [180] exhibited an infinite abelian square-free word $\mathbf{k}$ over a four-letter alphabet (there are no infinite abelian square-free words over a ternary alphabet). In this paper, we call this infinite abelian square-free word the *Keränen sequence*. Details about its construction can be found in [180, 181, 255].

Sequence Complexity.

The number of factors of a given size of an infinite word gives an intuitive notion of its *complexity* : a sequence is more complex (or richer) if it possesses a large number of different factors. We denote by $Fact_{\vec{\mathbf{z}}}(\ell)$ the number of factors of size ℓ of the sequence $\vec{\mathbf{z}}$.

F.3.2 Rivest's Proposals.

Keränen-DMD.

Rivest suggested to directly use the Keränen sequence as a source of dithering inputs. The dithering inputs are taken from the alphabet $\mathbf{a} = \{a, b, c, d\}$, and can be encoded by two bits. The number of data bits in the input of the compression function is thus reduced by only two bits, which improves the hashing efficiency. It is possible to generate the Keränen sequence online, one symbol at a time, in logarithmic space and constant amortized time.

Rivest's Concrete Proposal.

Rivest's concrete proposal is referred to as DMD-CP (Dithered Merkle-Damgård–Concrete Proposal). To speed up the generation of the dithering sequence, Rivest proposed a slightly modified scheme, in which the dithering symbols are 16-bit wide. If the message M is r blocks long, then for $1 \leq i < r$ the i-th dithering symbol has the form :

$$\left(0, \mathbf{k}[\left\lfloor i/2^{13} \right\rfloor], i \mod 2^{13}\right) \in \{0,1\} \times \mathbf{a} \times \{0,1\}^{13}$$

The idea is to increment the counter for each dithering symbol, and to shift to the next letter in the Keränen sequence, only when the counter overflows. This "diluted" dithering sequence can essentially be generated 2^{13} times faster than the Keränen sequence. The last dithering symbol has a different form :

$$(1, |M| \mod m) \in \{0,1\} \times \{0,1\}^{15}$$

F.4 Second Preimage Attacks on Dithered Merkle-Damgård

In this section, we present the first known second preimage attack on Rivest's dithered Merkle-Damgårdconstruction. In section F.4.1, we adapt the attack of section F.2 to Keränen-DMD, obtaining second preimages in time $(k + 39) \cdot 2^{n-k}$. We then apply the extended attack to MMD-CP, obtaining second preimages with about 2^{n-k+16} evaluations of the compression function. We show some examples of sequences which make the corresponding dithered constructions immune to our attack. This notably covers the case of Haifa [232]. Lastly, we present in section F.4.2 a variation of the attack, which includes an expensive preprocessing, but which is able to cope with sequences of high complexity over a small alphabet with a very small online cost.

F.4.1 Adapting the Attack to Dithered Merkle-Damgård

Let us now assume that the hashing algorithm uses a dithering sequence $\vec{\mathbf{z}}$. When building the collision tree, we must choose which dithering symbols to use. A simple solution is to use the same dithering symbol for all the edges at the same depth in the tree. A tuple of ℓ letters is then required to build the collision tree. We will also need an additional letter to connect the tree to the message M. This way, in order to build a collision tree of height ℓ, we have to fix a word ω of size $\ell + 1$, use $\omega[i]$ as the dithering symbol of depth i, and use the last letter of ω to realize the connection.

The dithering sequence makes the hash of a block dependent on its position in the whole message. Therefore, the collision tree can be connected to its target only at certain

positions, namely, at the positions where ω and $\vec{\mathbf{z}}$ match. The range of such an expandable message $\mathcal{M}$ is then given by :

$$\text{Range}\,\mathcal{M} = \Big\{ i \in \mathbb{N} \,\Big|\, (\ell + 1 \leq i) \wedge (\vec{\mathbf{z}}[i-\ell] \ldots \vec{\mathbf{z}}[i] = \omega) \Big\}.$$

Note that finding a connecting block B_0 in the second step defines the length of the instance of $\mathcal{M}$ that is required. If i_0 is in the range of $\mathcal{M}$, it will be possible to build this instance. Otherwise, another block B_0 has to be found.

To make sure that $\text{Range}\,\mathcal{M}$ is not empty, ω has to be a factor of $\vec{\mathbf{z}}$. Ideally, ω should be the factor of length $\ell + 1$ which occurs most frequently in $\vec{\mathbf{z}}$, as the cost of the attack ultimately depends on the number of connecting blocks tried before finding a useful one (with $i_0 \in \text{Range}\,\mathcal{M}$). What is the probability that a factor ω appears at a random position in $\vec{\mathbf{z}}$? Although this is highly sequence-dependent, it is possible to give a generic lower bound : in the worst case, all factors of size $\ell + 1$ appear in $\vec{\mathbf{z}}$ with the same frequency. In this setting, the probability that a randomly chosen factor of size $\ell + 1$ in $\vec{\mathbf{z}}$ is the word ω is $1/Fact_{\vec{\mathbf{z}}}(\ell + 1)$.

The main property of $\vec{\mathbf{z}}$ influencing the cost of our attack is its complexity (which is related to its min-entropy), whereas its repetition-freeness influences the cost of Kelsey and Schneier type attacks.

1. Choose the most frequent factor ω of $\vec{\mathbf{z}}$, with $|\omega| = \ell + 1$.
2. Build a collision tree of size ℓ using ω as the dithering symbols in all the leaf-to-root paths. Let h_T be the target value of the tree.
3. Find a connecting block B_0 mapping h_T to anyone of the h_i (say h_{i_0}), by using $\omega[\ell]$ as the dithering letter. Repeat until $i_o \in \text{Range}\,\mathcal{M}$.
4. Carry the remaining steps of the attack as described previously.

FIGURE F.2: Summary of the attack when a dithering sequence $\vec{\mathbf{z}}$ is used.

The cost of finding this second preimage for a given sequence $\vec{\mathbf{z}}$ is then :

$$2^{\frac{n}{2}+\frac{\ell}{2}+2} + Fact_{\vec{\mathbf{z}}}(\ell + 1) \cdot 2^{n-k+1} + 2^{n-\ell+1}.$$

Cryptanalysis of Keränen-DMD.

The cost of the extended attack against Keränen-DMD depends on the complexity of the sequence $\mathbf{k}$. Since it has a very regular structure, $\mathbf{k}$ has an unusually low complexity (there are other abelian square-free sequences with exponential complexity, see appendix F.6). In appendix F.7 we prove the following lemma :

Lemme F.1. *For $\ell \leq 85$, we have :*

$$Fact_{\mathbf{k}}(\ell) \leq 8 \cdot \ell + 332. \qquad \triangledown$$

By computing the exact number of factors of $\mathbf{k}$, it is possible to observe that this bound is tight (for example there are exactly 732 factors of size 50). With $\ell = 50$, and assuming that all factors occur equally often in $\mathbf{k}$, the probability that a connecting block falls in the range of $\mathcal{M}$ is $2^{-9.5}$. However, we observe that in the first 2^{30} symbols of $\mathbf{k}$ there are factors of size 50 occurring 2941800 times while some others only occur 910237 times. By choosing one of the former, and assuming that the frequency of this factor is the same

throughout **k**, the probability can be increased to $2^{-8.5}$, but we are not going to take this slight improvement into account when computing the cost of the attack.

Despite being strongly repetition-free, the sequence **k** offers an extremely weak security level against our attack. We illustrate this by evaluating the cost of our attack on Keranen-DMD :

$$2^{\frac{n}{2}+\frac{\ell}{2}+2} + (8 \cdot \ell + 340) \cdot 2^{n-k+1} + 2^{n-\ell+1}.$$

If n is greater than about $3k$, then by setting $\ell = k - 3$, the total cost of the attack is about :

$$(k + 39) \cdot 2^{n-k+4}$$

which is much smaller than 2^n in spite of the dithering.

Cryptanalysis of DMD-CP.

We now apply the attack to Rivest's concrete proposal. We first need to evaluate the complexity of its dithering sequence. Recall from section F.3.2 that it is based on the Keränen sequence, but that we move on to the next symbol of the sequence only when a 13 bit counter overflows. The original motivation was to reduce the cost of the dithering, but it has the unintentional effect of increasing the resulting sequence complexity. However, in appendix F.7 we prove that this effect is quite small :

Lemme F.2. *Let* **c** *denote the sequence obtained by diluting* **k** *with a 13-bit counter . Then for every* $0 \leq \ell < 2^{13}$*, we have :*

$$Fact_{\mathbf{c}}(\ell) = 8 \cdot \ell + 32760. \qquad \triangledown$$

The dilution does not generate a sequence of higher asymptotic complexity : it is still linear in ℓ, even though the constant term is larger due to the counter. The cost of the attack is therefore :

$$2^{\frac{n}{2}+\frac{\ell}{2}+2} + (8 \cdot \ell + 32768) \cdot 2^{n-k+1} + 2^{n-\ell+1}.$$

Again, if n is greater than about $3k$, the best value of ℓ is given by $\ell = k - 3$, and the complexity of the attack is then approximately :

$$(k + 4096) \cdot 2^{n-k+4} \simeq 2^{n-k+16}.$$

For settings corresponding to SHA-1, a second preimage can be computed in time 2^{121}.

Countermeasures.

Even though the dilution does not increase the asymptotic complexity of a sequence, the presence of a counter increases the complexity of the attack. If we simply used a counter over i bits as the dithering sequence, the number of factors of size ℓ would be $Fact(\ell) = 2^i$ (as long as $i \leq \ell$). The complexity of the attack would then become :

$$2^{\frac{n}{2}+\frac{\ell}{2}+2} + 2^{n-k+i+1} + 2^{n-\ell+1}.$$

In practice, the dominating term is $2^{n-k+i+1}$. By taking $i = k$, we would obtain a scheme which is resistant to our attack. This is essentially the choice made by the designers of HAIFA [232], but such a dithering sequence consumes k bits of bandwidth. Note that as long as the counter does not overflow, no variation of the attack of Kelsey and Schneier can be applied to the dithered construction.

A possible way to improve the resistance of Rivest's dithered hashing to this attack is to use a dithering sequence of high complexity over a small alphabet (to preserve bandwidth). In appendix F.6 we show that there is an abelian square-free sequence over 6 letters with complexity greater than $2^{\ell/2}$. Then, with $\ell = 2k/3$, the total cost of the online attack is $2^{n-2k/3}$.

Another possible way to repair Rivest's dithered hashing is to use a pseudo random sequence over a small alphabet. Even though it may not be repetition-free, its complexity is almost maximal. Suppose the alphabet has size $|\mathbf{a}| = 2^i$. Then the expected number of ℓ-letter factors in a pseudo random word of size 2^k is : $2^{i\cdot\ell} \cdot (1 - \exp -2^{k-i\cdot\ell})$ (refer to [136, 162] for a proof of this claim). The total optimal cost of the online attack is $2^{n-k/(i+1)+3}$ and is obtained with $\ell = k/(i+1)$. With 8-bit dithering symbols and if $k = 55$, as in the SHA family, the complexity of the attack is 2^{n-4}.

F.4.2 A Generic Attack on any Dithering Scheme With a Small Alphabet

The attacks described so far exploited the low complexity of Rivest's specific dithering sequences. In this section we show that the weakness is more general, and that after some preprocessing second preimages can be found in $\max(\mathcal{O}\left(2^k\right), \mathcal{O}\left(2^{(n-k)/2}\right))$ time for any dithering sequence (even of maximal complexity) if the dithering alphabet is small.

Outline of the Attack.

The new attack can be viewed as a type of time-memory tradeoff. For any given compression function, we precompute a fixed data structure called a *kite generator*[1] which can then be used to compute additional preimages for *any* dithering sequence and *any* given message of length $\mathcal{O}\left(2^k\right)$ in time $\max(\mathcal{O}\left(2^k\right), \mathcal{O}\left(2^{(n-k)/2}\right))$ and negligible additional space. Note that for the SHA-1 parameters of $n = 160$ and $k = 55$, the time complexity of the new attack is 2^{55}, which is just the time needed to hash the original message ! The size of the kite generator is $\mathcal{O}\left(|\mathbf{a}| \cdot 2^{n-k}\right)$ (where $|\mathbf{a}| = 4$ for Rivest's original proposal, and $|\mathbf{a}| = 2^{15}$ for Rivest's concrete proposal). The kite generator is a labelled directed graph whose 2^{n-k} vertices are labelled by some easily recognized subset of the chaining values that includes the IV (e.g., the tiny fraction of hash values which are extremely close to IV). Each directed edge (which can be traversed in both directions) is labelled by one letter α from the dithering alphabet and one message block m, and it leads from vertex h_1 to vertex h_2 if $F(h_1, m, a) = h_2$. Each vertex in the generator should have exactly two outgoing edges labelled by each dithering letter, and thus the *expected* number of ingoing edges labelled by each letter is also 2. The generator is highly connected in the sense that there is an exponentially large diverging binary tree with any desired dithering

1. We call it a kite generator since we use it generate *kites* of the form of

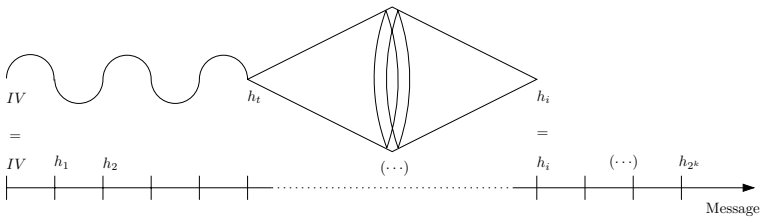

FIGURE F.3: A Kite

sequence starting at any vertex, and an exponentially large converging tree [2] with any desired dithering sequence (whose degrees are not always 2) ending at most vertices. It can be viewed as a generalization of the collision tree of Kelsey and Kohno [178], which is a single tree with a single root in only the converging direction and with no dithering labels.

Once computed (during an unbounded precomputation stage), we can use the generator to find a second preimage for any given message M with 2^k blocks and any dithering sequence. We first hash the long input M to find (with high probability) some intermediate hash value h_i which appears in the generator. We then use the generator to replace the first i blocks in the message by a different set of i blocks. We start from the generator vertex labelled by IV, and follow some path in the generator of length $i-(n-k)$ which has the desired dithering sequence (there are exponentially many paths we can choose from). It leads to some hash value h_t in the generator. We then evaluate the full diverging tree of depth $(n-k)/2$ and the desired dithering sequence starting at h_t, and the full converging tree of depth $(n-k)/2$ and the desired dithering sequence ending at h_i. Since the number of leaves in each tree is $\mathcal{O}\left(2^{(n-k)/2}\right)$ and they are labelled by only 2^{n-k} possible values, we expect by the birthday paradox to find a common chaining value among the two sets of leaves. We can now combine the long random chain of length $i-(n-k)$ with the two short tree chains of length $(n-k)/2$ to find a *kite*-shaped structure of the same length i and with the same dithering sequence as the original message between the two chaining values IV and h_i.

Finding Collisions inside a Kite.

The simplest way to find the common leaf is to store all the leaf values of the diverging and converging trees in an additional data structure, sort it, and look for repetitions. This requires time and space $\mathcal{O}\left(2^{(n-k)/2}\right)$. However, it is possible to find the common leaf value by a more sophisticated algorithm which requires the same time but negligible additional space. It is based on Pollard's ρ method, with appropriate modifications. Let $f_1(x)$ be the mapping from the root of the diverging tree to the leaf in which the path is determined by the desired dithering sequence where the first $n-k-1$ bits in x define which of the two outgoing edges with each dithering symbol to follow. Let $f_2(x)$ be a similar mapping from the root of the converging tree to the leaf specified by x. Finally, let $f_0(x)$ be the random function from $n-k$ bit values to $n-k$ bit values in which the last bit of x determines whether we apply f_1 or f_2 to x. By iterating f_0 on a random initial value x_0, we expect to enter a loop after $\mathcal{O}\left(2^{(n-k)/2}\right)$ steps. The entry point into this loop represents two different values x_1 and x_2 which converge under f_0 to the same value x_0. With constant probability x_1 and x_2 have different final bits, and in this case x_0 is a common leaf in the diverging and converging tree paths represented by $f_1(x_1)$ and $f_2(x_2)$ [3].

When the size of the dithering alphabet $\mathbf{a}$ exceeds 2^k (as in the HAIFAproposal) the size of the kite generator becomes larger than 2^n, and thus our attack becomes more expensive than the trivial attack that precomputes and stores one message that hashes to each one of the 2^n possible values.

The attack can be applied with essentially the same complexity even when the IV is not known during the precomputation stage (e.g., when it is time dependent). When we

2. See [106] for a formal justification of this claim.

3. Note that the length of the path in each one of the trees was extended from $(n-k)/2$ to $n-k-1$ in order to make sufficiently many leaves reachable from both roots, and thus we have to shorten the tail to $i-2(n-k-1)$ to get the same total path length.

hash the original long message, we have to find two intermediate hash values h_i and h_j (instead of IV and h_i), and connect them by a properly dithered kite-shaped structure of the same length.

Note that if we reduce the size of either the message or the kite generator, we are unlikely to find any common chaining values between the given message and the generator. Finding a way to connect the generator back into the message will require 2^{n-k+1} additional steps, and thus the time complexity of finding second preimages in arbitrarily dithered SHA-1 will jump from 2^{55} to at least 2^{106}. It is an interesting open problem whether we can precompute a smaller generator, and trade it off smoothly with a larger computing time without such quantum jumps.

F.5 An Attack on Shoup's UOWHF

In this section, we show that our attack is generic enough to be applied against hash functions enjoying a different security property, namely Universal One-Way Hash Function (UOWHF). A UOWHF is a keyed hash function H for which any computationally bounded adversary A wins the following game with negligible probability. First A chooses a message M, then a key K is chosen at random and given to A. The adversary wins if she violates the Target Collision Resistance (TCR) of H, that is if she generates a message M' different from M that collides with M for the key K (i.e. such that $H_K(M) = H_K(M')$ with $M \neq M'$).

At EUROCRYPT 2000, Shoup [270] had proposed a simple construction for a UOWHF that hashes messages of arbitrary size, given a UOWHF that hashes messages of fixed size. It is a Merkle-Damgård-like mode of operation, but before every iteration, one of several possible masks is XORed to the chaining value. The number of masks is logarithmic in the length of the hashed message, and the order in which they are used is carefully chosen to maximize the security of the scheme. This is reminiscent of dithered hashing, except that here the dithering process does not hit the bandwidth available to actual data.

We first describe briefly Shoup's construction, and then show how our attack can be applied against it. This way we prove that for this particular construction, Shoup's security bound is tight.

F.5.1 Description

The construction has some similarities with Rivest's dithered hashing. It starts from a universal one way compression function F that is keyed by a key K, $F_K\colon \{0,1\}^m \times \{0,1\}^n \to \{0,1\}^n$. This compression function is then iterated, as described below, to obtain a variable input length UOWHF H_K^F.

A certain number of "masks" are needed, each one of which is a random n-bit string. Assuming that the maximal size of hashed messages is 2^k then $k+1$ masks $M_0, \ldots, M_k$ are required. The key of the whole iterated function consists of K and of these masks. After each application of the compression function, a mask is XORed to the result. The order in which the masks are applied is defined by a specified sequence over the alphabet $\mathbf{a} = \{0, \ldots, k\}$. The scheduling sequence is $\vec{\mathbf{z}}[i] = \nu_2(i)$, for $1 \leq i \leq 2^k$, where $\nu_2(i)$ denotes the largest integer ν such that 2^ν divides i. Let M be a message that can be split into r blocks $x_1, \ldots, x_r$ and let h_0 be an arbitrary n-bit string. We define $h_i = F_K\left(h_{i-1} \oplus M_{\nu_2(i)}, x_i\right)$, and $H_K^F(M) = h_r$.

F.5.2 An Attack Matching the Security Bound

In [270], Shoup proves the following security result :

Théorème F.3 (Main result of [270]). *If an adversary is able to break the target collision resistance of H^F with probability ε in time T, then one can construct an adversary that breaks the target collision resistance of F in time T, with probability $\varepsilon/2^k$.* ♢

In this section we show that this bound is almost tight. First, we give an alternate definition of the dithering sequence $\vec{\mathbf{z}}$. We define :

$$u_i = \begin{cases} 0 & \text{if } i = 1, \\ u_{i-1}.(i-1).u_{i-1} & \text{otherwise.} \end{cases}$$

As an example, we have $u_4 = 010201030102010$. It is clear that $|u_i| = 2^i - 1$, and it is easy to show that for all i, u_i is a prefix of $\vec{\mathbf{z}}$. The dithering sequence is thus simply u_k (a prefix of $\vec{\mathbf{z}}$ of size $2^k - 1$ is arguably enough).

The most frequently-occurring factor of size $\ell < 2^k$ in $\vec{\mathbf{z}}$ is the prefix of size ℓ of $\vec{\mathbf{z}}$. It is a prefix of u_j with $j = \lceil \log_2(\ell + 1) \rceil$, and u_j itself occurs 2^{k-j} times in $\vec{\mathbf{z}} = u_k$. The probability for a random factor of $\vec{\mathbf{z}}$ of size ℓ to be exactly this candidate is equal to the number of occurrences of this candidate divided by the number of ℓ-bit strings in $\vec{\mathbf{z}}$. Thus this probability is $\frac{2^{k-j}}{2^k - \ell}$. This can in turn be lower-bounded by : $2^{-j} \geq \frac{1}{2(\ell+1)}$. Our attack can be applied against the TCR property of H^F as described above : just choose at random a (long) message x. Once the key is chosen at random, build a collision tree using a prefix of $\vec{\mathbf{z}}$ of size ℓ, and continue as described in section F.4. The cost of the attack is then :

$$T = 2^{\frac{n}{2} + \frac{\ell}{2} + 2} + 2(\ell + 1) \cdot 2^{n-k+1} + 2^{n-\ell+1}.$$

This attack breaks the target collision resistance with probability $\mathcal{O}(1)$. Therefore, with Shoup's result, one can construct an adversary A against F with running time $\mathcal{O}(T)$ and probability of success $\mathcal{O}\left(1/2^k\right)$. Therefore, the adversary $\mathbf{a}'$ which runs $\mathbf{a}$ 2^k times has a probability $\mathcal{O}(1)$ of breaking the TCR property. When $n \geq 3k$, $T \simeq (2k+3) \cdot 2^{n-k+1}$ (with $\ell = k - 1$), and thus the running time of the adversary is $\mathcal{O}\left(2^k \cdot T\right) = \mathcal{O}(k \cdot 2^n)$. If F is a random oracle, the best attack against F's TCR property runs in time $\mathcal{O}(2^n)$, which means that there is only a factor k between Shoup's security proof and our attack.

F.5.3 Further Improvement of the Attack

Our attack can be improved to obtain a gap $\mathcal{O}(\log k)$ between Shoup's security proof and our attack. To this end, we use the following trick. Instead of focusing on the most frequent word of size ℓ, we focus on the most frequent word of size $\mathcal{O}(\log_2(\ell))$ for which it is easier to connect a collision tree to the original message, because it occurs more frequently.

More precisely, let ω_0 be the $(s+1)$-letter prefix of $\vec{\mathbf{z}}$, where $s = \lceil \log_2(12\ell \cdot k) \rceil$. First we produce a collision tree for every ℓ-symbol word ω such that $\omega \| \omega_0$ is a factor of $\vec{\mathbf{z}}$. Thus, for every such word ω are associated a collision tree and the target h_ω of the collision tree. Since we are free to choose the hash value labelling the leaves of a tree, we now use the set of all these targets as the leaves for a new smaller collision tree using the s first letters of ω_0 as a dithering. This way we herd all the previously constructed collision tree in a bigger aggregated tree. Let h_T denote the target of the resulting collision tree. We continue the attack nearly as usual. We find a block B_0 such that

Parameters	n	k	Basic	Improved
SHA-1	160	55	2^{112}	2^{115}
SHA-256	256	55	2^{208}	2^{206}

$F_K\left(h_T \oplus M_{\omega_0[s+1]}, B_0\right) = h_{i_0}$ for some i_0, with $\vec{\mathbf{z}}[i_0-s-1]\dots\vec{\mathbf{z}}[i_0] = \omega_0$ and $i_0 \geq \ell+s+1$. This fixes a word $\omega_1 = \vec{\mathbf{z}}[i_0-s-\ell-3]\dots\vec{\mathbf{z}}[i_0-s-2]$ and the particular subtree generated with ω_1 as dithering symbols. Finally we generate an arbitrary prefix P of size $i_0-s-\ell-4$ whose hash is one of the leaf of the ω_1 subtree. Let T be the chain of $\ell+s$ blocks traversing successively the ω_1 subtree and then the small ω_0 collision tree, from h to h_T. The second preimage is therefore $P\|T\|B_0\|x_{i_0+1}\|\dots\|x_{2^k}$.

Let c_ℓ denote the number of ℓ-letter words w such $\omega\|\omega_0$ is a factor of $\vec{\mathbf{z}}$. Note that the previous attack is correct if and only if the number of leaves of the ω_0 collision tree is greater than c_ℓ the number of collision graph we have to herd. It can be shown that for the particular value chosen for s, $2^s \geq c_\ell$, thus the attack is correct and the cost of this attack is roughly :

$$c_\ell \cdot 2^{\frac{n}{2}+\frac{\ell}{2}+2} + 2^{\frac{n}{2}+\frac{\log_2(\ell)}{2}+2} + 2(1+\log_2\ell)\cdot 2^{n-k+1} + 2^{n-\ell+1}.$$

Compared to the previous attack, we now have to build c_ℓ collision trees of size ℓ instead of one, but it also becomes easier to connect our structure to the original message. Note that the term $2^{n/2+\log_2(\ell)/2+2}$ is always negligible before $c_\ell \cdot 2^{n/2+\ell/2+2}$.

If $n \gg 3k$, when $\ell = k$, the term $c_\ell \cdot 2^{n/2+\ell/2+2} = \mathcal{O}\left(2^{n/2+k/2+2\log_2 k}\right)$ is smaller than 2^{n-k+1} and the cost of the attack is $\mathcal{O}\left(\log k \cdot 2^{n-k}\right)$. Therefore, with the same proof as in the previous subsection, we can show that there is a factor $\mathcal{O}(\log k)$ between Shoup's security proof and our attack. Note that, depending on the parameters, this improved version of the attack may be worse than the basic version.

F.5.4 Comparing the Shoup and Rivest Dithering Techniques

An intriguing connection between Shoup's and Rivest's ideas shows up as soon as we notice that the scheduling sequence $\vec{\mathbf{z}}$ chosen by Shoup is abelian square-free. In fact, one year after Shoup's construction was published, Mironov [219] proved that an even stronger notion of repetition-freeness was necessary : $\vec{\mathbf{z}}$ is, and has to be, *even-free*. A word is even-free if all of its non-empty factors contain at least one letter an odd number of times. Note that all even-free words are abelian square-free. We believe that the role these non-trivial sequences play in iterated constructions in cryptography (such as hashing) has yet to be completely understood.

F.6 Appendix 1 : There are Abelian Square-Free Sequences of Exponential Complexity

Another way to repair DMD-CP would be to find an infinite abelian square-free sequence of exponential complexity. This is indeed possible, although we do not know how to do it without slightly enlarging the alphabet.

We start with the abelian square-free Keränen sequence $\mathbf{k}$ over $\{a,b,c,d\}$, and with another sequence $\mathbf{u}$ over $\{0,1\}$ that has an exponential complexity. Such a sequence can be built for example by concatenating the binary encoding of all the consecutive integers.

Then we can create a sequence $\tilde{\vec{\mathbf{z}}}$ over the union alphabet $\mathbf{a} = \{a, b, c, d, 0, 1\}$ by interleaving $\mathbf{k}$ and $\mathbf{u}$: $\tilde{\vec{\mathbf{z}}} = \mathbf{k}[1].\mathbf{u}[1].\mathbf{k}[2].\mathbf{u}[2]\ldots$. The resulting shuffled sequence inherits both properties : it is still abelian square-free, and has a complexity of order $\Omega\left(2^{\ell/2}\right)$.

Even with this exponentially complex dithering sequence, our attack is still more efficient than brute-force in finding second preimages. Although it may be possible to find square-free sequences with even higher complexity, it is probably very difficult to achieve optimal protection, and the generation of the dithering sequences is likely to become more and more complex.

F.7 Appendix 2 : Proofs of Sequence-Complexity Related Results

Sequences Generated by Morphisms.

We say that a function $\tau : \mathbf{a}^* \to \mathbf{a}^*$ is a *morphism* if for all words x and y, $\tau(x.y) = \tau(x).\tau(y)$. A morphism is then entirely determined by the images of the individuals letters. A morphism is said to be *r-uniform* (with $r \in \mathbb{N}$) if for all word x, $|\tau(x)| = r \cdot |x|$. If, for a given letter $\alpha \in \mathbf{a}$, we have $\tau(\alpha) = \alpha.x$ for some word x, then τ is *non-erasing* for α. Given a morphism τ and an initialization letter α, let u_n denote the n-th iterate of τ over α : $u_n = \tau^n(\alpha)$. If τ is r-uniform (with $r \geq 2$) and non-erasing for α, then u_n is a strict prefix of u_{n+1}, for all $n \in \mathbb{N}$. Let $\tau^\infty(\alpha)$ denote the limit of this sequence : it is the only fixed point of τ that begins with the letter α. Such infinite sequences are called *uniform tag sequences* [71] or *r-automatic* sequences [4].

Because they have a very regular structure, there is a spectacular result [71] regarding the complexity of infinite sequences generated by uniform morphisms :

Théorème F.4 (Cobham, 1972). *Let $\vec{\mathbf{z}}$ be an infinite sequence generated by an r-uniform morphism, and assume that the alphabet size $|\mathbf{a}|$ is constant. Then $\vec{\mathbf{z}}$ has* linear *complexity :*

$$Fact_{\vec{\mathbf{z}}}(\ell) \leq r \cdot |\mathbf{a}|^2 \cdot \ell. \qquad \diamondsuit$$

It is worth mentioning that similar results exist in the case of sequences generated by non-uniform morphisms [237, 101], although the upper bound can be quadratic.

Proofs.

Since the Keränen sequence is 85-uniform [180, 181, 255], the result of theorem F.4 gives : $Fact_{\mathbf{k}}(\ell) \leq 1360 \cdot \ell$. This upper-bound is relatively rough, and for particular values of ℓ, it is possible to obtain a much better approximation (which is tight) :

Lemme F.5 (of section F.4.1). *Let $\vec{\mathbf{z}}$ be an infinite sequence over the alphabet $\mathbf{a}$ generated by a r-uniform morphism τ. For all ℓ, $1 \leq \ell \leq r$, we have :*

$$Fact_{\vec{\mathbf{z}}}(\ell) \leq \ell \cdot \Big(Fact_{\vec{\mathbf{z}}}(2) - |\mathbf{a}|\Big) + \Big[(r+1) \cdot |\mathbf{a}| - Fact_{\vec{\mathbf{z}}}(2)\Big]. \qquad \triangledown$$

Démonstration. If $\ell \leq r$, then any factor of $\vec{\mathbf{z}}$ of size ℓ falls in one of these two classes (possibly both, but this does not matter, as we are looking for an upper bound) :

– Either it is a factor of $\tau(\alpha)$ for some letter $\alpha \in \mathbf{a}$. There are no more than $|A| \cdot (r - \ell + 1)$ such factors.

– Or it is a factor of $\tau(\alpha).\tau(\beta)$, for two letters $\alpha, \beta \in \mathbf{a}$ (and is not a factor of either $\tau(\alpha)$ or $\tau(\beta)$). For any given pair (α, β), there can only be $\ell - 1$ such factors. Moreover, $\alpha\beta$ must be a factor of size 2 of $\vec{\mathbf{z}}$.

So $Fact_{\vec{\mathbf{z}}}(\ell) \leq |A| \cdot (r - \ell + 1) + Fact_{\vec{\mathbf{z}}}(2) \cdot (\ell - 1)$. ∎

The result of this lemma can be specialized for the Keränen sequence, to obtain the result announced in section F.4.1. In this case, we have $r = 85$, $|\mathbf{a}| = 4$ and $Fact_{\mathbf{k}}(2) = 12$ (all non-repeating pairs of letters).

Lemme F.6 (of section F.4.1). *Let $\vec{\mathbf{z}}$ be an arbitrary sequence over $\mathbf{a}$, and let $\mathbf{d}$ denote the sequence obtained by diluting $\vec{\mathbf{z}}$ with a counter over i bits. Then for every ℓ not equal to 1 modulo 2^i, we have :*

$$Fact_{\mathbf{d}}(\ell) = (2^i - (\ell \mod 2^i) + 1) \cdot Fact_{\vec{\mathbf{z}}}(\lceil \ell \cdot 2^{-i} \rceil) \quad \text{(F.1)}$$

$$+ ((\ell \mod 2^i) - 1) \cdot Fact_{\vec{\mathbf{z}}}(\lceil (\ell - 1) \cdot 2^{-i} \rceil + 1) \quad \text{(F.2)}$$

▽

Démonstration. The counter over i bits splits the diluted sequence $\mathbf{c}$ into chunks of size 2^i (a new chunk begins when the counter reaches 0). In a chunk, the letter from $\vec{\mathbf{z}}$ does not change, and only the counter varies. To obtain the number of factors of size ℓ, let us slide a window of size ℓ over $\mathbf{d}$. This window overlaps at least $\lceil \ell \cdot 2^{-i} \rceil$ chunks (when the beginning of the window is aligned at the beginning of a chunk), and at most $\lceil (l - 1) \cdot 2^{-i} \rceil + 1$ chunks (when the window begins just before a chunk boundary). These two numbers are equal if and only if $\ell \equiv 1 \mod 2^i$. When this case is avoided, then these two numbers are consecutive integers.

This means that by sliding this window of size ℓ over $\mathbf{d}$ we will only observe factors of $\vec{\mathbf{z}}$ of size $\lceil \ell \cdot 2^{-i} \rceil$ and $\lceil \ell \cdot 2^{-i} \rceil + 1$. Given a factor of size $\lceil \ell \cdot 2^{-i} \rceil$ of $\vec{\mathbf{z}}$, there are $(2^i - (\ell \mod 2^i) + 1)$ positions of a window of size ℓ that allow us to observe this factor with different values of the counter (it essentially amounts to moving the window without crossing a chunk boundary).

Similarly, there are $((\ell \mod 2^i) - 1)$ positions of the window that contain a given factor of $\vec{\mathbf{z}}$ of size $\lceil \ell \cdot 2^{-i} \rceil + 1$. ∎

By taking $2 \leq \ell \leq 2^i$, we have that $\lceil \ell \cdot 2^{-i} \rceil = 1$. Therefore, only the number of factors of size 1 and 2 of $\vec{\mathbf{z}}$ come into play. The formula can be further simplified into :

$$Fact_{\mathbf{d}}(\ell) = \ell \cdot \Big(Fact_{\vec{\mathbf{z}}}(2) - Fact_{\vec{\mathbf{z}}}(1) \Big) + (2^i + 1) \cdot Fact_{\vec{\mathbf{z}}}(1) - Fact_{\vec{\mathbf{z}}}(2).$$

For the Keränen sequence with $i = 13$, this gives : $Fact_{\mathbf{d}}(\ell) = 8 \cdot \ell + 32760$.

Annexe G

Full Key-Recovery Attacks on HMAC/NMAC-MD4 and NMAC-MD5

CRYPTO 2007
[117] avec Gaëtan Leurent et Phong Q. Nguyen

Abstract : *At Crypto '06, Bellare presented new security proofs for HMAC and NMAC, under the assumption that the underlying compression function is a pseudo-random function family. Conversely, at Asiacrypt '06, Contini and Yin used collision techniques to obtain forgery and partial key-recovery attacks on HMAC and NMAC instantiated with MD4, MD5, SHA-0 and reduced SHA-1. In this paper, we present the first full key-recovery attacks on NMAC and HMAC instantiated with a real-life hash function, namely MD4. Our main result is an attack on HMAC/NMAC-MD4 which recovers the full MAC secret key after roughly* 2^{88} *MAC queries and* 2^{95} *MD4 computations. We also extend the partial key-recovery Contini-Yin attack on NMAC-MD5 (in the related-key setting) to a full key-recovery attack. The attacks are based on generalizations of collision attacks to recover a secret IV, using new differential paths for MD4.*

G.1 Introduction

Hash functions are fundamental primitives used in many cryptographic schemes and protocols. In a breakthrough work, Wang *et al.* discovered devastating collision attacks [290, 292, 293, 291] on the main hash functions from the MD4 family, namely MD4 [290], RIPE-MD [290], MD5 [292], SHA-0 [293] and SHA-1 [291]. Such attacks can find collisions in much less time than the birthday paradox. However, their impact on the security of existing hash-based cryptographic schemes is unclear, for at least two reasons : the applications of hash functions rely on various security properties which may be much weaker than collision resistance (such as pseudorandomness) ; Wang *et al.*'s attacks are arguably still not completely understood.

This paper deals with key-recovery attacks on HMAC and NMAC using collision attacks. HMAC and NMAC are hash-based message authentication codes proposed by Bellare, Canetti and Krawczyk [19], which are very interesting to study for at least three

reasons : HMAC is standardized (by ANSI, IETF, ISO and NIST) and widely deployed (*e.g.* SSL, TLS, SSH, Ipsec) ; both HMAC and NMAC have security proofs [18, 19] ; and both are rather simple constructions. Let H be an iterated Merkle-Damgård hash function. Its HMAC is defined by

$$\mathrm{HMAC}_k(M) = H(\bar{k} \oplus \mathrm{opad} \,||H(\bar{k} \oplus \mathrm{ipad} \,||M)),$$

where M is the message, k is the secret key, $\bar{k}$ its completion to a single block of the hash function, opad and ipad are two fixed one-block values. The security of HMAC is based on that of NMAC. Since H is assumed to be based on the Merkle-Damgård paradigm, denote by H_k the modification of H where the public IV is replaced by the secret key k. Then NMAC with secret key (k_1, k_2) is defined by :

$$\mathrm{NMAC}_{k_1,k_2}(M) = H_{k_1}(H_{k_2}(M)).$$

Thus, HMAC_k is essentially equivalent to $\mathrm{NMAC}_{H(k\oplus\mathrm{opad}),H(k\oplus\mathrm{ipad})}$ [1]. Attacks on NMAC can usually be adapted to HMAC (pending few modifications), except in the related-key setting [2].

HMAC/NMAC Security.

The security of a MAC algorithm is usually measured by the difficulty for an attacker having access to a MAC oracle to forge new valid MAC-message pairs. More precisely, we will consider two types of attack : the existential forgery where the adversary must produce a valid MAC for *a message of its choice*, and the universal forgery where the attacker must be able to compute the MAC of *any message*.

The security of HMAC and NMAC was carefully analyzed by its designers. It was first shown in [19] that NMAC is a pseudorandom function family (PRF) under the two assumptions that (A1) the keyed compression function f_k of the hash function is a PRF, and (A2) the keyed hash function H_k is *weakly collision resistant*. The proof for NMAC was then lifted to HMAC by further assuming that (A3) the key derivation function in HMAC is a PRF. However, it was noticed that recent collision attacks [290, 292, 293, 291] invalidate (A2) in the case of usual hash function like MD4 or MD5, because one can produce collisions for any public IV. This led Bellare [18] to present new security proofs for NMAC under (A1) only. As a result, the security of HMAC solely depends on (A1) and (A3). The security of NMAC as a PRF holds only if the adversary makes less than $2^{n/2}$ NMAC queries (where n is the MAC size), since there is a generic forgery attack using the birthday paradox with $2^{n/2}$ queries.

Since recent collision attacks cast a doubt on the validity of (A1), one may wonder if it is possible to exploit collision search breakthroughs to attack HMAC and NMAC instantiated with real-life hash functions. In particular, MD4 is a very tempting target since it is by far the weakest real-life hash function with respect to collision resistance. It is not too difficult to apply collision attacks on MD4 [290, 299], to obtain distinguishing and existential forgery attacks on HMAC/NMAC-MD4 : for instance, this was done independently by Kim *et al.* [184] and Contini and Yin [74]. The situation is more complex with MD5, because the differential path found in the celebrated MD5 collision attack [292] is not well-suited to HMAC/NMAC since it uses two blocks : Contini and Yin [74] turned instead to the much older MD5 pseudo-collisions of de Boer and Bosselaers [88] to obtain distinguishing

1. There is small difference in the padding : when we use $H(k||\cdot)$ instead of H_k, the length of the input of the hash function (which is included in the padding) is different.
2. If we need an oracle $\mathrm{NMAC}_{k_1,k_2+\Delta}$, we can not emulate it with an related-key HMAC oracle.

and existential forgery attacks on NMAC-MD5 in the related-key setting. It is the use of pseudo-collisions (rather than full collisions) which weakens the attacks to the related-key setting.

Interestingly, universal forgery attacks on HMAC and NMAC seem much more difficult to find. So far, there are only two works in that direction. In [74], Contini and Yin extended the previous attacks to *partial* key-recovery attacks on HMAC/NMAC instantiated with MD4, SHA-0, and a step-reduced SHA-1, and related-key *partial* key-recovery attacks on NMAC-MD5. In [252], Rechberger and Rijmen improved the data complexity of Kim *et al.* [184] attacks, and extended them to a partial key recovery against NMAC-SHA-1. These attacks are only partial in the sense that the NMAC attacks only recover the second key k_2, which is not sufficient to compute new MACs of arbitrary messages ; and the HMAC attacks only recover $H(k \oplus \text{ipad})$ where k is the HMAC secret key, which again is not sufficient to compute new MACs of arbitrary messages, since it does not give the value of k nor $H(k \oplus \text{opad})$. Note that recovering a single key of NMAC does not significantly improve the generic full key-recovery attack which recovers the keys one by one.

Very recently, Rechberger and Rijmen have proposed full key-recovery attacks against NMAC in the related-key setting in [253]. They extended the attack of [74] to a full key-recovery attack against NMAC-MD5, and introduced a full key-recovery attack against NMAC when used with SHA-1 reduced to 34 rounds.

Our Results.

We present what seems to be the first universal forgery attack, *without related keys*, on HMAC and NMAC instantiated with a real-life hash function, namely MD4. Our main result is an attack on HMAC/NMAC-MD4 which recovers the full NMAC-MD4 secret key after 2^{88} MAC queries ; for HMAC, we do not recover the HMAC-MD4 secret key k, instead we recover both $H(k \oplus \text{ipad})$ and $H(k \oplus \text{opad})$, which is sufficient to compute any MAC. We also obtain a full key-recovery attack on NMAC-MD5 in the related-key setting, by extending the attack of Contini and Yin [74]. This improvement was independently proposed in [252].

Our attacks have a complexity greater than the birthday paradox, so they are not covered by Bellare's proofs. Some MAC constructions have security proof against PRF-attacks, but are vulnerable to key-recovery attacks. For instance, the envelope method with a single key was proved to be secure, but a key-recovery attack using 2^{67} known text-MAC pairs, and 2^{13} chosen texts was found by Preneel and van Oorschot [250]. However, in the case of NMAC, we can prove that the security against universal forgery cannot be less than the security of the compression function against a PRF distinguisher (see Appendix G.6). This shows that NMAC offers good resistance beyond the birthday paradox : a universal forgery attack will have a time complexity of 2^n if there is no weakness in the compression function. Conversely, there is a generic attack against any iterated stateless MAC using a collision in the inner hash function to guess the two subkeys k_1 and k_2 independently, in the case of NMAC it requires $2^{n/2}$ queries and 2^{n+1} hash computations [249, 251].

Our attacks on MD4 and MD5 are rather different from each other, although both are based on IV-recovery attacks, which allow to recover the IV when one is given access to a hash function whose IV remains secret. Such IV-recovery attacks can be exploited to attack HMAC and NMAC because the oracle can in fact be very weak : we do not need the full output of the hash function ; essentially, we only need to detect if two related messages collide under the hash function. The MD5 related-key attack closely follows the Contini-Yin attack [74] : the IV-recovery attack is more or less based on message modification techniques. The MD4 IV-recovery attack is based on a new technique : we use differential

paths which depend on a condition in the IV. The advantage of this technique is that it can be used to recover the outer key quite efficiently, since we only need to control the *difference* of the inputs and not the *values* themselves. This part of the attack shares some similar ideas with [253]. To make this possible *without related keys*, we need a differential path with a message difference only active in the first input words. We found such IV-dependant paths using an automated tool described in [119]. To make this attack more efficient, we also introduce a method to construct cheaply lots of message pairs with a specific hash difference.

Our results are summarized in the following table, together with previous attacks where "Data" means online queries and "Time" is offline computations :

Attacks		Data	Time	Mem	Remark
Generic	E-Forgery	$2^{n/2}$	-	-	[251] Collision based
	U-Forgery	$2^{n/2}$	2^{n+1}	-	[251] Collision based
		1	$2^{2n/3}$	$2^{2n/3}$	[6] TM tradeoff, 2^n precomputation
NMAC-MD4	E-Forgery	2^{58}	-	-	[74] Complexity is actually lower [119]
HMAC-MD4	Partial-KR	2^{63}	2^{40}	-	[74] Only for NMAC
	U-Forgery	2^{88}	2^{95}	-	**New result**
NMAC-MD5	E-Forgery	2^{47}	-	-	[74]
	Partial-KR	2^{47}	2^{45}	-	[74]
Related keys	U-Forgery	2^{51}	2^{100}	-	**New result** – Same as [253]

Like [74], we stress that our results on HMAC and NMAC do not contradict any security proof ; on the contrary they show that when the hypotheses over the hash function are not met, an attack can be built.

Road Map.

This paper is divided in five sections. In Section G.2, we give background and notations on MD4, MD5 and collision attacks based on differential cryptanalysis. In Section G.3, we explain the framework of our key-recovery attacks on HMAC and NMAC, by introducing IV-recovery attacks. In Section G.4, we present key-recovery attacks on HMAC/NMAC-MD4. Finally, in Section G.5, we present related-key key-recovery attacks on NMAC-MD5.

G.2 Background and notation

Unfortunately, there does not seem to be any standard notation in the hash function literature. Here, we will use a notation similar to that of Daum [86].

G.2.1 MD4 and MD5

MD4 and MD5 follow the Merkle-Damgård construction. Their compression function are designed to be very efficient using 32-bit words and operations implemented in hardware in most processors :

– rotation $\lll$;
– addition mod 2^{32} $\boxplus$;

– bitwise boolean operations Φ_i. For MD4 and MD5, they are :
  - $\mathrm{IF}(x,y,z) = (x \wedge y) \vee (\neg x \wedge z)$
  - $\mathrm{MAJ}(x,y,z) = (x \wedge y) \vee (x \wedge z) \vee (y \wedge z)$
  - $\mathrm{XOR}(x,y,z) = x \oplus y \oplus z$
  - $\mathrm{ONX}(x,y,z) = (x \vee \neg y) \oplus z$.

  MD4 uses $\mathrm{IF}(x,y,z)$, $\mathrm{MAJ}(x,y,z)$ and $\mathrm{XOR}(x,y,z)$, while MD5 uses $\mathrm{IF}(x,y,z)$, $\mathrm{IF}(z,x,y)$, $\mathrm{XOR}(x,y,z)$ and $\mathrm{ONX}(x,y,z)$.

The compression function cMD4 (resp. cMD5) of MD4 (resp. MD5) uses an internal state of four words, and updates them one by one in 48 (resp. 64) steps. Their input is 128 bits $\times$ 512 bits, and their output is 128 bits. Here, we will assign a name to every different value of these registers, following [86] : the value changed on step i is called Q_i. Then the cMD4 compression function is defined by :

$$\begin{array}{|r|}\hline \text{Step update} : Q_i = (Q_{i-4} \boxplus \Phi_i(Q_{i-1}, Q_{i-2}, Q_{i-3}) \boxplus m_i \boxplus k_i) \lll s_i \\ \text{Input} : Q_{-4}||Q_{-1}||Q_{-2}||Q_{-3} \\ \text{Output} : Q_{-4} \boxplus Q_{44}||Q_{-1} \boxplus Q_{47}||Q_{-2} \boxplus Q_{46}||Q_{-3} \boxplus Q_{45} \\ \hline \end{array}$$

And the cMD5 compression function is given by :

$$\begin{array}{|r|}\hline \text{Step update} : Q_i = Q_{i-1} \boxplus (Q_{i-4} \boxplus \Phi_i(Q_{i-1}, Q_{i-2}, Q_{i-3}) \boxplus m_i \boxplus k_i) \lll s_i \\ \text{Input} : Q_{-4}||Q_{-1}||Q_{-2}||Q_{-3} \\ \text{Output} : Q_{-4} \boxplus Q_{60}||Q_{-1} \boxplus Q_{63}||Q_{-2} \boxplus Q_{62}||Q_{-3} \boxplus Q_{61} \\ \hline \end{array}$$

The security of the compression function was based on the fact that such operations are not "compatible" and mix the properties of the input.

We will also use $x^{[k]}$ to represent the $k+1$-th bit of x, that is $x^{[k]} = (x \ggg k) \bmod 2$ (note that we count bits and steps starting from 0).

G.2.2 Collision attacks based on differential cryptanalysis

It is natural to apply differential cryptanalysis to find collisions on hash functions based on block ciphers, like MD4 and MD5 (which both follow the Davies-Meyer construction). The main idea is to follow the differences in the internal state Q_i of the compression function, when the inputs (the IVs or the messages) have a special difference.

Our attacks on NMAC-MD5 are based on the MD5 pseudo-collision of de Boer and Bosselaers [88], like [74] (this is the only known attack against MD5 compression function's pseudo-randomness). Let IV be a 128-bit value satisfying the so-called dBB condition : the most significant bit of the last three 32-bit words of IV are all equal. Clearly, a randomly chosen IV satisfies the dBB condition with probability 1/4. It is shown in [88] that in such a case, a randomly chosen 512-bit message M satisfies with heuristic probability 2^{-46} :

$$\mathrm{cMD5}(IV, M) = \mathrm{cMD5}(IV', M),$$

where IV' is the 128-bit value derived from IV by flipping the most significant bit of each of the four 32-bit words of IV. The probability 2^{-46} is obtained by studying the most likely differences for the internal state Q_i, as is usual in differential cryptanalysis.

Our attacks on HMAC/NMAC-MD4 are based on recent collision search techniques for MD4 [290, 299], which are organized as follows :

1. A precomputation phase :
   – choose a message difference Δ
   – find a differential path
   – compute a set of sufficient conditions
2. Search for a message M satisfying all the conditions for a given IV ; then $\text{cMD4}(\text{IV}, M) = \text{cMD4}(\text{IV}, M \boxplus \Delta)$.

The differential path specifies how the computations of $\text{cMD4}(\text{IV}, M \boxplus \Delta)$ and $\text{cMD4}(\text{IV}, M)$ are related : it describes how the differences introduced in the message will evolve in the internal state Q_i. By choosing a special Δ with a low Hamming weight and extra properties, we can find differences in the Q_i which are very likely. Then we look at each step of the compression function, and we can express a set of sufficient conditions that will make the Q_i's follow the path. The conditions are on the Q_i's, and their values depends of the IV and the message M. For a given message M, we will have $\text{cMD4}(\text{IV}, M) = \text{cMD4}(\text{IV}, M \boxplus \Delta)$ if the conditions are satisfied ; we expect this to happen with probability 2^{-c} for a random message if there are c conditions. Wang introduced some further ideas to make the search for such message more efficient, but they can't be used in the context of NMAC because the IV is unknown.

G.3 Key-Recovery Attacks on HMAC and NMAC

In this section, we give a high-level overview of our key-recovery attacks on HMAC and NMAC instantiated with MD4 and MD5. Detailed attacks will be given in the next two sections : Section G.4 for MD4 and Section G.5 for MD5. We will assume that the attacker can request the MAC of messages of its choice, for a fixed secret key, and the goal is to recover that secret key. In the related-key setting, we will assume like in [74] that the attacker can request the MAC of messages of its choice, for the fixed secret key as well as for other related secret keys (with a chosen relation). In fact, we will not even need the full output of MAC requests : we will only need to know if the two MACs of messages of our choice collide or not.

To simplify our exposition, we will concentrate on the NMAC case :

$$\text{NMAC}_{k_1,k_2}(M) = H_{k_1}(H_{k_2}(M)).$$

The NMAC-MD4 attack can easily be extended to HMAC-MD4, pending minor modifications. Our NMAC attack will first recover k_2, then k_1. We will collect NMAC collisions of a special shape, in order to disclose hash collisions with first k_2 then k_1.

G.3.1 Extracting hash collisions from NMAC collisions

We will extract hash collisions from NMAC collisions, that is, pairs (M_1, M_2) of messages such that :

$$M_1 \neq M_2 \quad \text{and} \quad \text{NMAC}_{k_1,k_2}(M_1) = \text{NMAC}_{k_1,k_2}(M_2). \tag{C}$$

Our attacks are based on the elementary observation that H-collisions can leak through NMAC. More precisely, two messages M_1 and M_2 satisfy (C) if and only if they satisfy either (C1) or (C2) :

(C2) $M_1 \neq M_2$ and $H_{k_2}(M_1) = H_{k_2}(M_2)$: we have a collision in the inner hash function ;

(C1) $H_{k_2}(M_1) = N_1 \neq N_2 = H_{k_2}(M_2)$ and $H_{k_1}(N_1) = H_{k_1}(N_2)$: we have a collision in the outer hash function .

If we select M_1 and M_2 uniformly at random, then (C) holds with probability 2^{-128} if NMAC is a random function. However, if we select many pairs (M_1, M_2) in such a way that (C2) holds with a probability significantly higher than 2^{-128}, then whenever $\mathrm{NMAC}_{k_1,k_2}(M_1) = \mathrm{NMAC}_{k_1,k_2}(M_2)$, it will be likely that we also have (C2). More precisely, we have (since $Ci \cap C = Ci$) :

$$\frac{\Pr(C2|C)}{\Pr(C1|C)} = \frac{\Pr(C2)}{\Pr(C1)}$$

and we expect that $\Pr(C2) \gg \Pr(C1) \approx 2^{-128}$.

Note that the Merkle-Damgård construction used in H leads to a simple heuristic way to distinguish both cases (without knowing the secret keys k_1 and k_2) if M_1 and M_2 have the same length, and therefore the same padding block P : if $H_{k_2}(M_1) = H_{k_2}(M_2)$, then for any M, we have $H_{k_2}(M_1||P||M) = H_{k_2}(M_2||P||M)$. In other words, the condition (C2) is preserved if we append $P||M$ to both M_1 and M_2 for a randomly chosen M, but that is unlikely for the condition (C1).

To illustrate our point, assume that we know a non-zero Δ such that for all keys k_2, a randomly chosen one-block message M_1 satisfies with probability 2^{-64} the condition $H_{k_2}(M_1) = H_{k_2}(M_2)$ where $M_2 = M_1 \boxplus \Delta$. If we select 2^{64} one-block messages M_1 uniformly at random and call the NMAC oracle on each M_1 and $M_1 \boxplus \Delta$, we are likely to find a pair $(M_1, M_2 = M_1 \boxplus \Delta)$ satisfying (C). By the previous reasoning, we expect that such a pair actually satisfies (C2).

Thus, the NMAC oracle allows us to detect collisions on H_{k_2}, if we are able to select messages which have a non-negligible probability of satisfying (C2). To detect collisions in H_{k_1}, we will use the values of k_2 (recovered using collisions in H_{k_2}) : then, we can compute H_{k_2} and directly check whether the NMAC collision come from (C1). We now explain how to use such collision detections to recover the secret keys k_2 and k_1.

G.3.2 IV-recovery attacks

The previous subsection suggests the following scenario. Assume that a fixed key k is secret, but that one is given access to an oracle which on input M_1 and M_2, answers whether $H_k(M_1) = H_k(M_2)$ holds or not. Can one use such an oracle to recover the secret key k ? If so, we have what we call an IV-recovery attack.

An IV-recovery attack would clearly reveal the second key k_2 of NMAC, because of (C2). But it is not clear why this would be relevant to recover the outer key k_1. To recover k_1 thanks to (C1), we would need the following variant of the problem. Namely, one would like to retrieve a secret key k_1 when given access to an oracle which on input M_1 and M_2, answers whether $H_{k_1}(H_{k_2}(M_1)) = H_{k_1}(H_{k_2}(M_2))$ holds or not, where k_2 is known. Since the messages are first processed through a hash function, the attacker no longer chooses the input messages of the keyed hash function, and this oracle is much harder to exploit than the previous one. We call such attacks composite IV-recovery attacks. In the attack on HMAC/NMAC-MD4, we will exploit the Merkle-Damgård structure of H_{k_2} to efficiently extend the basic IV-recovery attacks into composite IV-recovery attacks.

We will present two types of IV-recovery attacks. The first type is due to Contini and Yin [74] and uses related messages, while the second type is novel, based on IV-dependent differential paths.

Using related messages.

We present the first type of IV-recovery attacks. Assume that we know a specific differential path corresponding to a message difference Δ and with total probability p much larger than 2^{-128}. In other words, a randomly chosen message M will satisfy with probability p :

$$H_k(M) = H_k(M \boxplus \Delta).$$

By making approximately $2/p$ queries to the H_k-oracle, we will obtain a message M such that $H_k(M) = H_k(M \boxplus \Delta)$. Contini and Yin [74] then make the heuristic assumption that the pair $(M, M \boxplus \Delta)$ must follow the whole differential path, and not just the first and last steps. Since they do not justify that assumption, let us say a few words about it. The assumption requires a strong property on our specific differential path : that there are no other differential paths with better (or comparable) probability. In some sense, differential cryptanalysis on block ciphers use similar assumptions, and to see how realistic that is, one makes experiments on reduced-round versions of the block cipher. However, one might argue that there are intuitively more paths in hash functions than in block ciphers because of the following facts :

- the message length of a compression function is much bigger than the length of the round key in a block cipher.
- a step of the compression function is usually much simpler than a block-cipher step.

Also, because the paths for hash functions have a different shape from those of block ciphers, experiments on reduced-round versions may not be as conclusive.

The paper [] shows that for usual differential paths (like those of MD4), if $(M, M \boxplus \Delta)$ satisfies the whole path, then one can build plenty of messages M^* closely related to M such that :

- If a specific internal register Q_i (during the computation of $H_k(M)$) satisfies certain conditions, then the pair $(M^*, M^* \boxplus \Delta)$ follows the whole path with probability p or larger, in which case $H_k(M^*) = H_k(M^* \boxplus \Delta)$.
- Otherwise, the pair $(M^*, M^* \boxplus \Delta)$ will drift away from the path at some position, and the probability of $H_k(M^*) = H_k(M^* \boxplus \Delta)$ is heuristically 2^{-128}.

Thus, by sending to the oracle many well-chosen pairs $(M', M' \boxplus \Delta)$, one can learn many bits of several internal register Q_i's during the computation of $H_k(M)$. Applying exhaustive search on the remaining bits of such Q_i's, one can guess the whole contents of four consecutive Q_i's. By definition of cMD4 and cMD5, it is then possible to reverse the computation of $H_k(M)$, which discloses $k = (Q_{-4}, Q_{-3}, Q_{-2}, Q_{-1})$.

Using IV-dependent differential paths.

We now present a new type of IV-recovery attacks, that we will apply against MD4. Assume again that we know a specific differential path corresponding to a message difference Δ and with total probability p much larger than 2^{-128}, but assume this time that the path is IV-dependent : it holds only if the IV satisfies a specific condition (SC). In other words, if k satisfies (SC), then a randomly chosen message M will satisfy with probability p :

$$H_k(M) = H_k(M \boxplus \Delta).$$

But if k does not satisfy (SC), the pair $(M, M \boxplus \Delta)$ will drift away from the differential path from the first step, leading us to assume that $H_k(M) = H_k(M \boxplus \Delta)$ will hold with probability only 2^{-128}.

This would lead to the following attack : we would submit approximately $2/p$ pairs $(M, M \boxplus \Delta)$ to the H_k-oracle, and conclude that k satisfies (SC) if and only if $H_k(M) =$

$H_k(M \boxplus \Delta)$ for at least one M. When sufficient information on k has been gathered, the rest of k can be guessed by exhaustive search if we have at least one collision of the form $H_k(M) = H_k(M \boxplus \Delta)$.

Notice that in some sense the differential path of the MD5 pseudo-collision [88] is an example of IV-dependent path where (SC) is the dBB condition, but it does not disclose much information about the IV. We would need to find many IV-dependent paths. Our attack on HMAC/NMAC-MD4 will use 22 such paths, which were found by an automated search.

We note that such attacks require an assumption similar to the previous IV-recovery attack. Namely, we assume that for the same message difference Δ, there is no differential paths with better (or comparable) probability, with or without conditions on the IV. To justify this assumption for our HMAC/NMAC-MD4 attack, we have performed experiments which will be explained in Section G.4.

G.3.3 Subtleties between the inner and outer keys

Although the recovery of the inner key k_2 and the outer key k_1 both require IV-recovery attacks, we would like to point out subtle differences between the two cases. As mentioned previously, the recovery of k_2 only requires a basic IV-recovery attack, while the recovery of k_1 requires a composite IV-recovery attack. The composite IV-recovery attacks will be explained in Section G.4 for MD4, and Section G.5 for MD5.

When turning a basic IV-recovery attack into a composite IV-recovery, there are two important restrictions to consider :

- We have no direct control over the input of the outer hash function, it is the result of the inner hash function. So IV-recovery attacks using message modifications will become much less efficient when turned into composite IV-recovery. We will see this in Section G.5 when extending the partial key-recovery from [74] into a full key-recovery.
- Since the input of H_{k_1} is a hash, its length is only 128 bits. Any differential path using a message difference Δ with non-zero bits outside these 128 first bits will be useless. This means that the partial key-recovery attacks from [74] against MD4, SHA-0 and reduced SHA-1 can't be extended into a full key-recovery.
  Using related keys one can use a differential path with a difference in the IV and no message difference – such as the one from [88] – and try a given message with both keys. However, if we want to get rid of related keys, we need a differential path with no IV difference and a difference in the beginning of the message.

G.3.4 Summary

To summarize, our attacks will have essentially the following structure (the MD5 attack will be slightly different because of the related-key setting) :

1. Apply an IV-recovery attack to retrieve k_2, repeating sufficiently many times :
   (a) Select many one-block messages M uniformly at random.
   (b) Observe if $\mathrm{NMAC}_{k_1,k_2}(M) = \mathrm{NMAC}_{k_1,k_2}(M \boxplus \Delta_1)$ for some M and a well-chosen Δ_1.
   (c) Deduce information on k_2.
2. Apply a composite IV-recovery attack to retrieve k_1, repeating sufficiently many times :
   (a) Construct carefully many pairs (M_1, M_2).

(b) Observe if $\mathrm{NMAC}_{k_1,k_2}(M_1) = \mathrm{NMAC}_{k_1,k_2}(M_2)$ for some pair (M_1, M_2).

(c) Deduce information on k_1.

G.4 Attacking HMAC/NMAC-MD4

G.4.1 Our IV-recovery Attack against MD4

In order to find differential paths which leak information about the key, we consider differential paths with a message difference in the first word (eg. $\delta m_0 = 1$). Then in the first steps of the compression function, we have :

$$Q_0 = (Q_{-4} \boxplus \mathrm{IF}(Q_{-1}, Q_{-2}, Q_{-3}) \boxplus m_0) \lll 3$$
$$Q'_0 = (Q_{-4} \boxplus \mathrm{IF}(Q_{-1}, Q_{-2}, Q_{-3}) \boxplus m_0 \boxplus 1) \lll 3$$

So $Q_0^{[3]} \neq Q_0'^{[3]}$. Then

$$Q_1 = (Q_{-3} \boxplus \mathrm{IF}(Q_0, Q_{-1}, Q_{-2}) \boxplus m_1) \lll 7$$
$$Q'_1 = (Q_{-3} \boxplus \mathrm{IF}(Q'_0, Q_{-1}, Q_{-2}) \boxplus m_1) \lll 7$$

Thus, if $Q_{-1}^{[3]} \neq Q_{-2}^{[3]}$, we will have $\mathrm{IF}(Q_0, Q_{-1}, Q_{-2}) \neq \mathrm{IF}(Q'_0, Q_{-1}, Q_{-2})$ and $Q_1 \neq Q'_1$. On the other hand, if $Q_{-1}^{[3]} = Q_{-2}^{[3]}$ and there is no carry when going from Q_0 to Q'_0, then $Q_1 = Q'_1$. Therefore, collision paths where $Q_{-1}^{[3]} = Q_{-2}^{[3]}$ will be significantly different from collision paths where $Q_{-1}^{[3]} \neq Q_{-2}^{[3]}$. This suggests that the collision probability will be correlated with the condition (SC) : $Q_{-1}^{[3]} = Q_{-2}^{[3]}$, and we expect to be able to detect the bias. More precisely we believe that the case $Q_{-1}^{[3]} \neq Q_{-2}^{[3]}$ will give a much smaller collision probability, since it means that an extra difference is introduced in step 1.

To check this intuition experimentally, we ran one cMD4 round (16 steps) with a random IV on message pairs $(M, M \boxplus 1)$ where M was also picked at random, and we looked for pseudo-collisions in $Q_{12}...Q_{15}$ with the following properties :

– The weight of the non-adjacent form of the difference is lower or equal to 4.

– There is no difference on $Q_{12}^{[12]}$.

The second condition is required to eliminate the paths which simply keep the difference introduced in $Q_0^{[3]}$ without modifying it. We ran this with $5 \cdot 10^{11}$ random messages and IVs and found 45624 collisions out of which 45515 respected the condition : this gives a ratio of about 420. This does not prove that we will have such a bias for collisions in the full MD4, but it is a strong evidence.

The same arguments apply when we introduce the message difference in another bit k (*ie.* $\delta m_0 = 2^k$) : we expect to find more collisions if $Q_{-1}^{[k \boxplus s_0]} = Q_{-2}^{[k \boxplus s_0]}$.

We ran a differential path search algorithm to find such paths, and we did find 22 paths for different values of k with $Q_{-1}^{[k \boxplus s_0]} = Q_{-2}^{[k \boxplus s_0]}$. The path for $k = 0$ is given in Appendix **??**, and the other paths are just a rotation of this one. The corresponding set of sufficient conditions contains 79 conditions on the internal variables Q_i, so we expect that for a random message M :

$$\begin{aligned} \Pr\left[\mathrm{MD4}(M) = \mathrm{MD4}(M + \Delta)\right] = p \geq 2^{-79} \qquad & \text{if } Q_{-1}^{[k \boxplus s_0]} = Q_{-2}^{[k \boxplus s_0]} \\ \ll p \qquad & \text{if } Q_{-1}^{[k \boxplus s_0]} \neq Q_{-2}^{[k \boxplus s_0]} \end{aligned}$$

If we try 2^{82} message pairs per path, we will find a collision for every path whose condition is fulfilled with a probability [3] of more than 99%. Then we know 22 bits of the IV ($Q_{-1}^{[k \boxplus s_0]} = Q_{-2}^{[k \boxplus s_0]}$ or $Q_{-1}^{[k \boxplus s_0]} \neq Q_{-2}^{[k \boxplus s_0]}$), which leaves only 2^{106} IV candidates. To check if a given IV is the correct one, we just check whether it gives a collision on the pairs colliding with the real IV, so we expect to find the IV after computing 2^{105} pairs of hashes in an offline phase.

We show in Appendix G.7.2 how to reduce the search space to 2^{94} keys by extracting more than one bit of information when a collision is found. This gives an IV-recovery attack against MD4 with a data complexity of 2^{88} MD4 oracle queries, and a time complexity of 2^{94} MD4 evaluations.

G.4.2 Deriving a Composite IV-recovery Attack against MD4

To turn this into a composite IV-recovery attack, we need to efficiently compute message pairs M, M' such that $H_{k_2}(M) = H_{k_2}(M') \boxplus \Delta$ (we will need 2^{82} such pairs). As we know k_2 (in the HMAC attack, we first recover it using the basic IV-recovery attack), we can compute $H_{k_2}(M)$ and find such pairs offline. If we do this naively using the birthday paradox, we need to hash about 2^{106} random messages to have all the pairs we need [4]. Then we can use the IV-recovery attack to get k_1.

Actually, we can do much better : we use the birthday paradox to find one pair of one-block messages (R, R') such that $H_{k_2}(R') = H_{k_2}(R) \boxplus \Delta$, and then we extend it to a family of two-block message pairs such that $H_{k_2}(R'||Q') = H_{k_2}(R||Q) \boxplus \Delta$ with very little extra computation. In the end, the cost to generate the messages with $H_{k_2}(M) = H_{k_2}(M') \boxplus \Delta$ will be negligible and the composite IV-recovery attack is as efficient as the basic one. This is the most important part of our work : thanks to our new path, we only need a low level of control on the input of the hash function to extract the IV.

Extending a Pair of Good Messages into a Family of Pairs.

We will shows how we will create many message pairs with $H_{k_2}(M) = H_{k_2}(M') \boxplus \Delta$. We use a pair of one-block message (R, R') such that $H_{k_2}(R') = H_{k_2}(R) \boxplus \Delta$. Then we will generate a second block pair (Q, Q') such that $H_{k_2}(R'||Q') \boxminus H_{k_2}(R||Q) = \Delta$. Thanks to the Davies-Meyer construction of the compression function, all that we need is a difference path which starts with a Δ difference and ends with a zero difference in the internal state ; then the feed-forward of H will keep $\delta H = \Delta$. This path can be found by a differential path search algorithm, or created by hand by slightly modifying a collision path.

In this step, we also have to take care of the padding in MD4. Usually we ignore it because a collision at the end of a block is still a collision after an extra block of padding, but here we want a specific non-zero difference, and this will be broken by an extra block. So we have to adapt our collision finding algorithm to produce a block with a 55-byte message M and the last 9 bytes fixed by the MD4 padding. This can be done with nearly the same complexity as unconstrained MD4 collisions (about 4 MD4 computations per collision) using the technique of Leurent [203]. Thus, the cost of the message generation in the composite IV-recovery attack drops from 2^{106} using the birthday paradox to 2^{90} and becomes negligible in the full attack.

3. We have $\left(1 - \left(1 - 2^{-79}\right)^{2^{82}}\right)^{22} > 0.992$

4. This gives 2^{210} pairs of messages, and each pair has a probability of 2^{-128} to have the correct difference.

G.4.3 MD4 Attack Summary

This attack uses the same IV-recovery attack for the inner key and the outer key, with a complexity of 2^{88} online queries and 2^{94} offline computations. We manage to keep the complexity of the composite IV-recovery as low as the basic IV-recovery because we only need to control the hash differences, and we introduce a trick to generate many messages with a fixed hash difference.

In Appendix G.7.1 we show how to reduce a little bit the query complexity of the attack, and in the end the NMAC full key-recovery attack requires 2^{88} requests to the oracle, and 2×2^{94} offline computations.

G.5 Attacking NMAC-MD5

In this section, we will describe the attack of Contini and Yin [74], and we extend it to a full key recovery. This improved attack was independently found by Rechberger and Rijmen in [253].

As for MD4, the IV-recovery attack is based on a specific differential path, and assumes that when a collision is found with the given message difference, the Q_i's follow the path. This gives some bits of the internal state already, and a kind of message modification technique to disclose more bits is proposed in [74].

If the path depends on the value of a bit $Q_t^{[k]}$ in the step t, then we can extract some extra bits from the register Q_t : set $\Delta = 2^{k-1}$ and modify the message M into a message M^* :

$$m_j^* = \begin{cases} m_j & \text{if } j < t \\ m_j + \Delta & \text{if } j = t \\ \text{random} & \text{if } j > t \end{cases}$$

Then, using MD4 or MD5 step update without the rotation, we have :

$$Q_j^* = \begin{cases} Q_j & \text{if } j < t \\ Q_j + \Delta & \text{if } j = t \\ \text{random} & \text{if } j > t \end{cases}$$

We call the oracle with enough such messages (with a different random part) to distinguish if Q_t^* still follows the path. If it does, this means that $Q_t^{*[k]} = Q_t^{[k]}$: there was no carry in $Q_t + \Delta$, therefore $Q_t^{[k-1]} = 0$. On the other hand, if it does not follow the path anymore, then $Q_t^{[k-1]} = 1$.

We can find $Q_t^{[k-2]}$ if we set Δ so that there is a carry up to the bit k if and only if $Q_t^{[k-2]} = 1$:

$$\Delta = \begin{cases} 2^{k-2} & \text{if } Q_j^{[k-1]} = 1 \\ 2^{k-2} + 2^{k-1} & \text{if } Q_j^{[k-1]} = 0 \end{cases}$$

and we can repeat this to get the bits $Q_t^{[k-1]} ... Q_t^{[0]}$.

The rotation will have little effect over this simplified explanation, it will mainly limit the number of bits we can recover (see [74] for details).

G.5.1 The IV-recovery Attack against MD5

The IV-recovery attack on MD5 is the same as the one presented in [74]. It uses the related-message technique with the pseudo-collision path of de Boer and Bosselaers [88].

Since the differences are in the IV and not in the message, the IV-recovery needs an oracle that answers whether $\mathrm{MD5}_{\mathrm{IV}}(M) = \mathrm{MD5}_{\mathrm{IV}'}(M)$, instead of the standard oracle that answers whether $\mathrm{MD5}_{\mathrm{IV}}(M) = \mathrm{MD5}_{\mathrm{IV}}(M')$. To apply this to an HMAC key-recovery, we will have to use the related-key model : we need an oracle for NMAC_{k_1,k_2}, $\mathrm{NMAC}_{k'_1,k_2}$ and NMAC_{k_1,k'_2}.

The IV-recovery attack in the related-key setting requires 2^{47} queries and 2^{45} hash computations, and this translates into a partial key-recovery (we will recover k_2) against NMAC-MD5 in the related-key model with the same complexity.

G.5.2 Deriving a Composite IV-recovery against MD5

To extend this to a composite IV-recovery attack, we run into the problem previously mentioned ; to use this attack we need to create many inputs N^* of the hash function related to one input N, but these inputs are the outputs of a first hash function, and we cannot choose them freely : $N = \mathrm{MD5}_{k_2}(M)$. However, we know k_2, so we can compute many $N_R = H_{k_2}(R)$ for random messages R and select those that are related to a particular N ; if we want to recover bits of Q_t we will have to choose $32(t+1)$ bits of N_R. We also run into the problem that any N_R is only 128 bits long ; the last 384 bits will be fixed by the padding and there are the same for all messages. Therefore, we can only use the related-message technique to recover bits of the internal state of in the very first steps, whereas in the simple IV-recovery it is more efficient to recover the internal state of later steps (Contini and Yin used step 11 to 14). If we want to recover bits of Q_0 (due to the rotation we can only recover 25 bits of them), we need to produce 24×2^{45} messages N^* with the first 32 bits chosen ; this will cost $24 \times 2^{45} \times 2^{32} \approx 2^{82}$ hash computations. Then, we know 25 bits of Q_0, plus the most significant bit of Q_1, Q_2, and Q_3 ; we still have 100 bits to guess. Thus, we have a related-key composite IV-recovery attack against MD5 with $2 \times 24 \times 2^{45} \approx 2^{51}$ oracle queries and 2^{100} MD5 evaluations.

If we try to guess bits in Q_1, we have to select at least 2^{44} hashes with 64 chosen bits ; this costs about 2^{108} MD5, so it does not improve the attack.

G.5.3 MD5 Attack Summary

Thus, the Contini-Yin NMAC-MD5 attack can be extended into a full key-recovery attack in the related-key setting, with a query complexity of 2^{51}, a time complexity of 2^{100} MD5 operations, and success rate of 2^{-4} (due to the dBB condition for k_1 and k_2).

It is a very simple extension of the attack from Contini and Yin : we apply their technique to recover the outer key, but since we cannot choose the value of $H_{k_2}(M)$, we compute it for many random messages until we find a good one. This requires to change the step in which we extract internal bits, and the complexity become much higher.

Acknowledgement

Part of this work is supported by the Commission of the European Communities through the IST program under contract IST-2002-507932 ECRYPT, and by the French government through the Saphir RNRT project.

G.6 Appendix 1 : NMAC Security beyond the Birthday Paradox

In this section we study the security of NMAC when the attacker can make enough queries to use the birthday paradox. There is a generic existential forgery attack, but we will show that universal forgery against NMAC requires more resources. Actually, we will prove a much stronger statement : NMAC is secure against short forgery (less than one block) if the underlying compression function is a PRF.

We define the 1-block MAC advantage of a MAC-adversary as (using the same notations as [18], B is the set of 1-block messages) :

$$\mathbf{Adv}_f^{\text{1-MAC}}(A) = \Pr\left[\texttt{pad}(M) \in B,\ (M,x) \text{ is a forgery} : (M,x) \leftarrow A^{f(K,\cdot),VF_f(K,\cdot,\cdot)}; K \xleftarrow{\$} Keys\right]$$

Note that the attacker is allowed to make any query of any length to the MAC oracle, we only limit the length of the forgery. We require that the padded message fit in one block, so the actual message has to be somewhat smaller ; for instance, in the case of MD4/MD5, the forged message has to be 447 bits or less.

The only generic short forgery attack we are aware of is the key-recovery attack using the birthday paradox which require $2^{n/2}$ queries and a time of 2^{n+1} (the generic forgery attack against iterated MAC produces at least a two-block forgery). On the other hand, short message forgeries are possible if the compression function is weak : using the partial key-recoveries attacks of Contini and Yin to find k_2, one can compute a pair of short messages (M_1, M_2) such that $H_{k_2}(M_1) = H_{k_2}(M_2)$ and forge a MAC for one of them. This pair can be found using the birthday paradox in time $2^{n/2}$, or in the case of MD4, in time about 2^9 using the results of [203] to include the padding in the one-block message.

Obviously, this kind of attack is stronger than a PRF-attack against HMAC, and weaker than a universal forgery or a key-recovery. The following theorem proves that such short forgeries are as hard as a PRF attack against the compression function, *ie.* it requires a time of the order of 2^n if there is no weakness in the compression function.

Théorème G.1. *Let A_{NMAC} be a 1-block MAC-adversary against NMAC instantiated with the compression function h. Then, there exist PRF-adversaries A_1 and A_2 against h such that :*

$$\mathbf{Adv}_{\text{NMAC}}^{\text{1-MAC}}(A_{\text{NMAC}}) \leq \mathbf{Adv}_h^{\text{prf}}(A_1) + \mathbf{Adv}_h^{\text{prf}}(A_2) + 2^{-n+1}$$

Furthermore, A_1 and A_2 have the same time complexity as A_{NMAC}, and just make one extra oracle query. ♢

Démonstration. We will build the adversary A_1 from A_{NMAC} by simulating the NMAC oracle. We are given a function g, and we try to guess if g is a random function or $h(K,\cdot)$ for some K. First we choose a random k_2, and we will then answer NMAC queries with $g(H_{k_2}(\cdot))$. A_{NMAC} will output some message M and a tag x, and we guess that g is a $f(K,\cdot)$ iff $x = g(H_{k_2}(M))$. Thus, we have :

- If g was a random function, $x = g(H_{k_2}(M))$ holds with a very low probability :
  - If we did not query the oracle g with the input $H_{k_2}(M)$ yet, then $g(H_{k_2}(M))$ is random and it will be equal to x with probability 2^{-n}.
  - If M collides under H_{k_2} with one of the q NMAC queries made by A_{NMAC}, this means we can build a PRF-attacker A_2 against h from A_{NMAC} : we answer NMAC queries using a random oracle for H_{k_1} and the given g' to compute the inner hash function (it will require one call to the oracle g' and some computations of f), then

we output 1 iff $H_{k_2}(M) \in Q$, where Q denotes the set of H_{k_2} outputs computed by A_2. If g was a random function, there is a negligible probability that we answer 1, and if g if some $h(K,\cdot)$ we will recognize it whenever A_{NMAC} succeeds to forge. Hence :

$$\begin{aligned}\Pr\left[A_2^{\$} \Rightarrow 1\right] &\leq 2^{-n}\\ \Pr\left[A_2^{f(K,\cdot)} \Rightarrow 1\right] &\geq \Pr\left[A_1^{\$} \Rightarrow 1, f(K,M) \in Q\right]\\ \Pr\left[A_1^{\$} \Rightarrow 1, f(K,M) \in Q\right] &\leq \mathbf{Adv}_h^{\text{prf}}(A_2) + 2^{-n}\end{aligned}$$

By combining these two cases, we have :

$$\begin{aligned}\Pr\left[A_1^{\$} \Rightarrow 1\right] &= \Pr\left[A_1^{\$} \Rightarrow 1, f(K,M) \in Q\right] + \Pr\left[A_1^{\$} \Rightarrow 1, f(K,M) \notin Q\right]\\ &\leq \mathbf{Adv}_h^{\text{prf}}(A_2) + 2^{-n} + 2^{-n}\end{aligned}$$

– If g is $h(K,\cdot)$, we will correctly output 1 if A_{NMAC} makes a correct forgery, and we finish the proof by combining the two cases :

$$\begin{aligned}\Pr\left[A_1^{f(K,\cdot)} \Rightarrow 1\right] &\geq \mathbf{Adv}_{\text{NMAC}}^{1-\text{MAC}}(A_{\text{NMAC}})\\ \mathbf{Adv}_{\text{NMAC}}^{1-\text{MAC}}(A_{\text{NMAC}}) &\leq \mathbf{Adv}_h^{\text{prf}}(A_1) + \mathbf{Adv}_h^{\text{prf}}(A_2) + 2^{-n+1}\end{aligned}$$

■

G.7 Appendix 2 : Improving the MD4 IV-recovery

G.7.1 Reducing the Online Cost

First, we can easily lower the number of calls to the NMAC-oracle in the first phase of the IV-recovery. Instead of trying 22×2^{82} random message pairs, we will choose the messages more cleverly so that each message belongs to 22 pairs : we first choose 490 bits of the message at random and then use every possibility for the 22 remaining bits. Thus, we only need 2^{83} calls to the oracle instead of 22×2^{83}.

Note that we cannot use this trick in the composite IV-recovery attack, so the number of queries for the full key-recovery will only be halved (the queries for the basic IV-recovery for k_2 become negligible compared to the queries for the composite IV-recovery that will reveal k_2).

G.7.2 Reducing the Offline Cost

We may also lower the computational cost of the attack, by getting more than one bit of the IV once a collision has been found. This will require the extra assumption the colliding messages follow the differential path in step 1 (previously we only needed step 0), but this seems quite reasonable, for the same reasons. Out of the 22 paths used to learn IV bits, let p be the number of paths for which the condition holds, and a collision is actually found. From each message that collides following the differential path, we can also extract some conditions on the internal states Q_0 and Q_1. These states are not part of the IV, but since we know the message used, we can use these conditions to learn something on the IV. If we have a message pair that collides with $M' \boxminus M = 2^k$, we will call them $M^{(k)}$ and $M'^{(k)}$ and the condition gives us $Q_0^{[k \boxplus s_0]}(M^{(k)})$ and $Q_1^{[k \boxplus s_0]}(M^{(k)})$. The idea is the following (the symbol '➤' summarizes the number of bits to guess at each step) :

1. We guess Q_{-1}. Let $n = 32 - |Q_{-1}|$ be the number of `0` bits in Q_{-1} (we use $|x|$ to denote the Hamming weight of x).
2. We compute 22 bits of Q_{-2} using the conditions on the IV, and we guess the others ➤ 10 bits.
3. We guess the bits of Q_{-3} used to compute Q_0. Since we have $Q_0 = (Q_{-4} \boxplus \mathrm{IF}(Q_{-1}, Q_{-2}, Q_{-3}) \boxplus k_0 \boxplus m_0) \lll s_0$, we only need $Q_{-3}^{[i]}$ when $Q_{-1}^{[i]} = 0$ ➤ n bits.
4. We have $Q_{-4} = (Q_0 \ggg s_0) \boxminus \mathrm{IF}(Q_{-1}, Q_{-2}, Q_{-3}) \boxminus m_0 \boxminus k_0$. If we use it with the message $M^{(0)}$ and take the equation modulo 2, it becomes : $Q_{-4}^{[0]} = Q_0^{[s_0]}(M^{(0)}) \boxminus (\mathrm{IF}(Q_{-1}, Q_{-2}, Q_{-3}) \boxminus m_0^{(0)} \boxminus k_0) \bmod 2$, and it gives us $Q_{-4}^{[0]}$.

   Then, if we write the equation with $M^{(1)}$ and take it modulo 2, we will learn $Q_0^{[s_0]}(M^{(1)})$ from $Q_{-4}^{[0]}$. Since we know $(Q_0(M^{(1)}) \lll s_0) \bmod 4$ from $Q_0^{[s_0]}(M^{(1)})$ and $Q_0^{[1 \boxplus s_0]}(M^{(1)})$, we can take the equation modulo 4 to learn $Q_{-4}^{[1]}$.

   By repeating this process, we learn the full Q_{-4}, but we need to guess the bit i when we don't have a message pair $M^{(i)}, M'^{(i)}$ ➤ $32 - p$ bits.
5. We apply the same process to compute the remaining bits of Q_{-3}. We already know n bits and we expect to be able to compute a ratio of $p/32$ of the missing ones. ➤ $\frac{(32-n)(32-p)}{32}$ bits.

So, for each choice of Q_{-1} in step 1, we have to try a number of choices for the other bits that depends on the Hamming weight $32 - n$ of Q_{-1}. In the end, the number of keys to try is :

$$\sum_{Q_{-1}} 2^{10+n+32-p+(32-n)(32-p)/32} = 2^{74-2p}\left(1 + 2^{p/32}\right)^{32}$$

With $p = 11$, this becomes a little less than 2^{90}, but the complexity depends on the number of conditions fulfilled by the key. If we assume that every condition has a probability of one half to hold, we can compute the average number of trials depending on the keys, and we will have to try half of them :

$$\frac{1}{\#k}\sum_{k} 2^{74-2p}\left(1 + 2^{p/32}\right)^{32} < 2^{93.8}$$

Hence, we have an IV-recovery attack requiring less than 2^{88} queries to the NMAC oracle, and less than 2^{94} offline hash computations. See the full version of this paper for a detailed complexity analysis.

G.7.3 Complexity Details

First, we will have to compute sums of Hamming weight power :

$$\begin{aligned}
S_n(\alpha) &= \sum_{x \in \mathbb{Z}_{2^n}} \alpha^{|x|} \\
S_1(\alpha) &= \alpha^0 + \alpha^1 && = 1 + \alpha \\
S_{n+1}(\alpha) &= S_n(\alpha) + \alpha S_n(\alpha) && = (1+\alpha)S_n(\alpha) \\
S_n(\alpha) &= (1+\alpha)^n
\end{aligned}$$

This allows us to compute the total complexity for every Q_{-1} :

$$\begin{aligned}\sum_{Q_{-1}} 2^{10+n+32-p+(32-n)(32-p)/32} &= 2^{74-2p} \sum_{Q_{-1}} 2^{np/32} \\ &= 2^{74-2p} S_{32}\left(2^{p/32}\right) \\ &= 2^{74-2p}\left(1+2^{p/32}\right)^{32}\end{aligned}$$

At the end, we need to compute the average complexity over the keys : we will reduce it to a sum only over the 22 bits of k related to the conditions counted in p.

$$\begin{aligned}\frac{1}{\#k}\sum_{k} 2^{74-2p}\left(1+2^{p/32}\right)^{32} &= \frac{2^{74}}{2^{22}} \sum_{k' \in \mathbb{Z}_{2^{22}}} 2^{-2|k'|}\left(1+2^{|k'|/32}\right)^{32} \\ &= 2^{52} \sum_{k' \in \mathbb{Z}_{2^{22}}} 2^{-2|k'|} \sum_{i=0}^{32} \binom{32}{i} 2^{|k'|i/32} \\ &= 2^{52} \sum_{i=0}^{32} \binom{32}{i} S_{22}\left(2^{i/32-2}\right) \\ &= 2^{52} \sum_{i=0}^{32} \binom{32}{i}\left(1+2^{i/32-2}\right)^{22} \\ &< 2^{93.8}\end{aligned}$$

Attaques logicielles et matérielles

Annexe H :

The Doubling Attack - *hy Upwards Is Better than Downwards*, CHES 2003
Pierre-Alain Fouque et Frédéric Valette
Cet article présente une attaque par collision sur l'implémentation de l'algorithme Square-and-Multiply qui lit les bits du haut vers le bas, c'est-à-dire des bits de poids fort vers les bits de poids faible. Nous avons remarqué que pendant le chiffrement du message M et du message M^2, certaines valeurs étaient identiques quand d_i valait 0 et différentes sinon. Ceci nous a permis de montrer que deux contremesures proposées par Coron pour éviter les attaques différentielles n'étaient pas sûres.

Annexe I :

Attacking Unbalanced RSA-CRT Using SPA, CHES 2003
Pierre-Alain Fouque, Gwenaëlle Martinet et Guillaume Poupard
Cet article présente une attaque contre l'implémentation de Garner pour calculer efficacement la recombinaison par le théorème des restes chinois. Cette attaque permet dans le cas des signatures RSA avec un module utilisant deux facteurs déséquilibrés de retrouver la factorisation de N en regardant le temps de calcul ou la consommation de puissance d'envion 90,000 signatures.

Annexe J :

Power Attack on Small RSA Public Exponent, CHES 2006
Pierre-Alain Fouque, Sébastien Kunz-Jacques, Gwenaëlle Martinet, Frédéric Muller et Frédéric Valette
Cet article présente une attaque pour retrouver une clé privée RSA quand l'exposant public est petit et qu'une implémentation laisse fuir des informations sur l'exposant. Si l'exposant est randomisé pour éviter les attaques différentielles, alors il est possible d'obtenir plusieurs informations sur les exposants. Comme ces exposants sont reliés à la clé privée, nous avons montré comment les utiliser pour retrouver la clé privée en entier.

Annexe H

The Doubling Attack- *Why Upwards is Better than Downwards*

CHES 2003
[128] avec Frédéric Valette

Abstract : *The recent developments of side channel attacks have lead implementers to use more and more sophisticated countermeasures in critical operations such as modular exponentiation, or scalar multiplication in the elliptic curve setting. In this paper, we propose a new attack against a classical implementation of these operations that only requires two queries to the device. The complexity of this so-called "doubling attack" is much smaller than previously known ones. Furthermore, this approach defeats two of the three countermeasures proposed by Coron at CHES '99.*

H.1 Introduction

Modular exponentiation or scalar multiplication are the main parts of the most popular public key cryptosystems such as RSA [257] or DSA [227]. This very sensitive operation can be efficiently implemented in smart cards products. However, data manipulated during this computation should often be kept secret, so the implementation of such algorithms must be protected against side channel attacks. For example, during the generation of an RSA signature by a device, the secret exponent is used to transform a message related data into a digital signature via modular exponentiation.

Timings and power attacks, initially presented by Kocher [195, 196] are now well studied and various countermeasures have been proposed. Those attacks represent a real threat when we consider operations that both involve secret data and require a long computation time. The consequence is that naive implementation of RSA based or discrete log based cryptosystems usually leak information about the secret key.

In this paper we present a new side channel attack, that we called "doubling attack", which allows to recover the secret scalar used in the binary scalar multiplication or the secret exponent used in the binary exponentiation algorithm. It is worth to notice that contrary to previous attacks which work for the "Left-to-Right" and the "Right-to-Left"

implementations of the binary modular exponentiation, the doubling attack only works for the "Left-to-Right" implementation. Furthermore, the "Left-to-Right" implementation is often used since it requires only one variable. The new attack enables to recover the secret key decryption of RSA [257] or the key decryption of ElGamal [133]. It can also be used to obtain the secret key of the Diffie-Hellman authentication system. We only focus on the decryption cases. In this attack we assume that the adversary mounts a chosen ciphertext attack. This is a valid assumption in a side channel scenario, since randomized paddings avoiding chosen ciphertext attack, such as OAEP [24], are checked after the running of the decryption process. As a consequence, the binary exponentiation or multiplication is always performed and side channel attacks can be mounted on these algorithms. The attack on the RSA cryptosystem is a direct application of the doubling attack. On discrete-log based cryptosystems, we describe the attack in the elliptic curve setting, since the doubling attack allows to defeat classical countermeasures which are mainly proposed to elliptic curve systems.

In this paper, we first remind classical binary scalar multiplication algorithms. Then, we shortly describe different types of side channel attacks such as simple power analysis and differential power analysis but also the attack of Messerges, Dabbish and Sloan [217] in order to motivate the most frequently used countermeasures.

Then, we present an improvement of Messerges *et al* attack that applies when so-called downward algorithms are used. It has a much smaller complexity since it only requires two queries to the device in order to recover all the secret data. This new attack is called "*doubling attack*" since it is based on the doubling operation in the elliptic curve setting. We also explain how to use this new attack to defeat Coron's countermeasures [76].

H.2 Binary scalar multiplication algorithms

In classical cryptosystems based on the RSA or on the discrete logarithm problem, the main operation is modular exponentiation. In the elliptic curve setting, the corresponding operation is the scalar multiplication. From an algorithmic point of view, those two operations are very similar ; the only difference is the underlying group structure. In this paper, we consider operations over a generic group, without using any additional property. The consequence is an immediate application to the elliptic curve setting but it should be clear that all what we state can be easily transposed to modular exponentiation.

Scalar multiplication is usually performed using the "double-and-add" method that computes $d \times P$ using the binary representation of the scalar $d = \sum_{i=0}^{n} d_i \times 2^i$:

$$d \times P = \sum_{i=0}^{n} d_i \times \left(2^i \times P\right)$$

Two versions of the double-and-add algorithm are usually considered, according to the order of the terms in the previous sum. The first routine starts from the most significant bit and works downward. This method is usually called "Left-to-Right" (see figure 12).

The second routine starts from the least significant bit and works upward. This method is also known as "Right-to-Left" (see figure 13).

The first implementation is the most frequently used since it requires less memory. Up to now, no distinction was made on the security of those routines since all proposed attacks can be adapted to both implementations. In the following sections, we focus on the downward implementations and we show that it may be much more easily attacked than upward versions.

Algorithme 12 Downward "Left-to-Right" double-and-add(P,d)

$S \leftarrow 0$
for i from n down to 0 **do**
    $S = 2.S$
    if $d_i = 1$ **then**
        $S = S + P$
    end if
end for
return S

Algorithme 13 Upward "Right-to-Left"double-and-add(P,d)

$S \leftarrow 0$
$T = P$
for i from 0 to n **do**
    if $d_i = 1$ **then**
        $S = S + T$
        $T = 2.T$
    end if
end for
return S

H.3 Power Analysis Attacks

It is well known that naive double-and-add algorithms are subject to power attacks introduced by Kocher *et al* [196]. More precisely, they introduced two types of power attacks : Simple Power Analysis (SPA) and Differential Power Analysis (DPA) we now shortly remind.

H.3.1 Simple Power Analysis

The first type of attack consists in observing the power consumption in order to guess which instruction is executed. For example, in the previous algorithm, one can easily recover the exponent $d = \sum_{i=0}^{n} d_i 2^i$ by distinguishing the doubling from the addition instruction. To avoid this attack, downward double-and-add algorithm is usually modified using so-called "dummy" instructions (see figure 14).

Algorithme 14 Downward double-and-add(P,d) resistant against SPA

$S[0] \leftarrow 0$
for i from n down to 0 **do**
    $S[0] = 2.S[0]$
    $S[1] = S[0] + P$
    $S[0] = S[d_i]$
end for
return $S[0]$

Although this new algorithm is immune to SPA, a more sophisticated treatment of power consumption measures can still enable to recover the secret scalar d.

H.3.2 Differential Power Analysis

DPA uses power consumption to retrieve information on the operand of the instruction. More precisely, it no longer focuses on which instruction is executed but on the Hamming weight of the operands used by the instruction. Such an attack has been described in the elliptic curve setting in [76, 231].

This technique can also be used in a different way. Messerges, Dabbish and Sloan introduced "Multiple Exponent Single Data" attack [217]. Note that, for our purpose, a better name would be "Multiple Scalar Single Data". We first assume that we have two identical equipments available with the same implementation of algorithm 14, one with an unknown scalar d and another one with a chosen scalar e. In order to discover the value of d, using correlation between power consumption and operand value, we can apply the following algorithm. We guess the bit d_n of d which is first used in the double-and-add algorithm and we set e_n to this guessed value. Then, we compare the power consumption of the two equipments doing the scalar multiplication of the same message. If the consumption is similar during the two first steps of the inner loop, it means that we have guessed the correct bit d_n. Otherwise, if the consumption differs in the second step, it means that the values are different and that we have guessed the wrong bit. So, after this measure, we know the most significant bit of d. Then, we can improve our knowledge on d by iterating this attack to find all bits as it is illustrated in the algorithm of figure 15.

Algorithme 15 MESD attack to find secret scalar d

```
for i from 0 to n do
    e_i = 0
end for
for i from 0 to n do
    e_{n-i} = 1
    choose M randomly
    double-and-add(P,d) on equipment 1
    double-and-add(P,e) on equipment 2
    if no correlation at step (i + 1) then
        e_{n-i} = 0
    end if
end for
return e
```

This kind of attack is well known and some classical countermeasures are often implemented. For example, the Chaum's blinding technique [69] can be used to protect an RSA implementation since it prevents an attacker from knowing the data used in the exponentiation. This method cannot be applied directly for computations based on the Discrete Logarithm problem as there is no public exponent associated with the secret exponent, as in RSA.

H.4 Usual DPA countermeasures for double-and-add algorithm

The most well know countermeasures for scalar multiplication on elliptic curve have been published by Coron [76]. In this paper, the author describes three different countermeasures which are respectively based on the blinding of the scalar, on the blinding of the

point or on the blinding of the multiplication. We now recall the description of the first two countermeasures. Then we explain, in section H.5, how to defeat them.

H.4.1 Coron's first countermeasure

During the computation of a scalar multiplication, this scalar can be blinded by adding a multiple of the number $\mathcal{E}$ of points of the curve. For this purpose, the algorithm needs a random value r which length is fixed to 20 bits in [76]. Then, the algorithm computes $(d + r\mathcal{E})P$ which is obviously equal to dP. This countermeasure, depicted in figure 16, is very efficient since the scalar value changes for each computation.

Algorithme 16 Implementation 1 secure against DPA

pick random value r
$d' = d + r\mathcal{E}$
return double-and-add(P,d')

H.4.2 Coron's second countermeasure

The second solution is based on the same idea as Chaum's blind RSA signature scheme. However, since we use a discrete log based problem, applying this method requires twice the time needed for a single scalar multiplication. To be more efficient, it is proposed in [76] to store a secret point R and the associated value $S' = dR$. The multiplication of P by d is performed by computing $d(P + R)$ and then subtracting S' to the result. The variability is obtained by doubling R and S' at each execution as shown in figure 17.

Algorithme 17 Implementation 2 secure against DPA

pick $b \in \{0, 1\}$ at random
$R = (-1)^b.2.R$
$S' = (-1)^b.2.S'$
$S =$ double-and-add$(P + R, d)$
return $S - S'$

In this paper we will not focus on the third countermeasure proposed by Coron since it has been partially broken by Goubin in [142]. Our attack does not work on this countermeasure and does not allow to enhance Goubin's attack.

These two countermeasures are well admitted to be efficient against power attacks. Other recently proposed countermeasures, such as randomized NAF [145] or [159, 277, 160] mainly focus on improving efficiency in terms of speed.

H.5 The new attack

We introduce a new attack mainly based on two reasonable assumptions. This attack is able to recover the secret scalar with a few requests to the card. The adversary needs to send chosen messages directly to the double-and-add algorithm. Indeed, when considering decryption of the ElGamal cryptosystem or of the RSA cryptosystem for instance, the padding can only be verified at the end of the computation.

The idea of the attack is based on the fact that, even if an adversary is not able to tell which computation is done by the card, he can at least detect when the card does twice the same operation. More precisely, if the card computes $2.A$ and $2.B$, the attacker is not

able to guess the value of A nor B but he is able to check if $A = B$. Such an assumption is reasonable since this kind of computation usually takes many clock cycles and depends greatly on the value of the operand. This assumption has been used in a stronger variant and validated by Schramm *et al.* in [266]. Indeed, they are able to distinguish collisions during one DES round computation which is much more difficult than distinguishing collisions during a doubling operation. If the noise is negligible, a simple comparison of the two power consumption curves during the doubling will be sufficient to detect this equality.

If the noise is more important we propose two solutions to detect equalities. The first and easiest one is to compare the average of several consumptions curves with A and B. However asking twice the same computation may be impossible. In that case, a better solution is to use the tiny differences on many clock cycles since a point doubling usually takes a few thousand cycles. By summing the square of the differences between these curves on each clock cycle, we can decrease the influence of noise. This approach is precised in appendix H.7.

H.5.1 Description of the doubling attack

The so-called "*doubling attack*" is based on the fact that similar intermediate values may be manipulated when working with points P and $2P$. However this idea only works when using the downward routine.

Let us first consider an example. Let $d = 78 = 64 + 8 + 4 + 2$, i.e. $n = 6$ and

$$(d_0, d_1, d_2, d_3, d_4, d_5, d_6) = (0, 1, 1, 1, 0, 0, 1)$$

Then we compare the sequence of operations when the downward binary scalar multiplication algorithm of figure 14 is used to compute $d \times P$ (on the left) and $d \times (2P)$ (on the right) :

i	d_i	comput. of dP	comput. of $d(2P)$
6	1	2×0	2×0
		$0 + P$	$0 + 2P$
5	0	$2 \times P$	$2 \times 2P$
		$2P + P$	$4P + 2P$
4	0	$2 \times 2P$	$2 \times 4P$
		$4P + P$	$8P + 2P$
3	1	$2 \times 4P$	$2 \times 8P$
		$8P + P$	$16P + 2P$
2	1	$2 \times 9P$	$2 \times 18P$
		$18P + P$	$36P + 2P$
1	1	$2 \times 19P$	$2 \times 38P$
		$38P + P$	$76P + 2P$
0	0	$2 \times 39P$	$2 \times 78P$
		$78P + P$	$156P + 2P$
		return $78P$	return $156P$

If we focus on the doubling operations, we notice that some of them manipulate the same operand. More precisely, we observe that the doubling operation at rank i in the computation of dP is the same as the doubling operation at rank $i-1$ in the computation of $d(2P)$ if and only if $d_{i-1} = 0$. Consequently, all the bits (except the least significant

one) can be deduced from the SPA analysis of only two power consumption curves, the first one obtained during the computation of dP and the second one with $d(2P)$.

More formally, let us denote the partial sums $S_k(P) = \sum_{i=0}^{i=k} d_{n-i}2^{k-i} \times P$. This value is the content of $S[0]$ in the algorithm 14 after $k+1$ iterations. Therefore we also refer to $S_k(P)$ as an intermediate result of the binary scalar multipliation algorithm. So this value is used in the next doubling operation in any case. Besides

$$\begin{aligned} S_k(P) &= \sum_{i=0}^{k} d_{n-i}2^{k-i} \times P \\ &= \sum_{i=0}^{k-1} d_{n-i}2^{k-1-i} \times (2P) + d_{n-k} \times P \\ &= S_{k-1}(2P) + d_{n-k} \times P \end{aligned}$$

Thus the intermediate result of the downward double-and-add algorithm with P at step k will be equal to the intermediate result with $2P$ at step $k-1$ if and only if d_{n-k} is null.

Using the same example as before, we obtain :

value of step k	0	1	2	3	4	5	6
value of d_{n-k}	1	**0**	**0**	1	1	1	**0**
value of $S_k(P)$	P	**2P**	**4P**	$9P$	$19P$	$39P$	**78P**
value of $S_k(2P)$	**2P**	**4P**	$8P$	$18P$	$38P$	**78P**	$156P$

In conclusion, we just need to compare the doubling computation at step $k+1$ for P and at step k for $2P$ to recover the bit d_{n-k}. If both computations are identical, d_{n-k} is equal to 0 otherwise d_{n-k} is equal to 1. This can also be observed by shifting the second measurement curve by one step to the right and comparing it to the first curve. Therefore, with only two requests to the card, it is possible to recover all the bits of the secret scalar.

Note that this attack also works with addition-subtraction chains such as Non Adjacent Form representation [222]. It allows to recover all the zeros in the NAF coding which represent roughly two third of the bits according to the paper of Morain and Olivos [222]. The missing information can be recovered by exhaustive search or by a more efficient method such as an adaptation of the baby step giant step algorithm for short Diffie-Hellman exponent. Indeed, if the prime group order is a 160-bit prime number, then only 54 bits remain to be discovered. Moreover, the Baby Step Giant Step algorithm can be used to reduce the complexity of the discovery of the discrete logarithm to 2^{27} in time and in memory.

H.5.2 Application of the doubling attack to Coron's countermeasures

The first countermeasure of Coron uses a 20-bit random value to blind the secret scalar at each request to the card. This size of random value is sufficient to resist usual DPA attacks. However it is not enough to resist our new doubling attack. Indeed, due to the birthday paradox, after 2^{10} requests with P and 2^{10} requests with $2P$, there should exist a common scalar with high probability. In order to recover this collision on the scalar, the attacker needs to compare each curve obtained with P to each curve obtained with $2P$. With the method described before, assuming the same scalar is used, it is possible to find the position of zeroes in this scalar. The right pair will be distinguished as it will give a scalar with enough zeroes. Indeed, if the corresponding scalars are different, it is unlikely

that common intermediate values appear on both computations. Hence identifying the right pair is quite easy and requires only 2^{20} comparisons. In case several bits of the scalar cannot be clearly identified due to the noise, they can be recovered by exhaustive search. The attack is summarized in figure 18

Algorithme 18 Attack of Coron's first countermeasure

while no correct pair is found **do**
request computation with P and store measurement $C(P)$ in set $\mathcal{A}$
request computation with $2P$ and store measurement $C(2P)$ in set $\mathcal{B}$
compare $C(P)$ with all set $\mathcal{B}$
compare $C(2P)$ with all set $\mathcal{A}$
if two measurements have many common intermediate squaring, a correct pair is found
end while
exhaustive search for undefined bits of the scalar recovered with the correct pair.

If the number of undefined bits is too large, one can notice that the number of correct pairs increases as the square of the number of extra requests. With about 2^{15} requests in each group, the number of correct pairs will be approximately 2^{10} which may help to decrease the work of the exhaustive search.

The second countermeasure is even more vulnerable to the doubling attack. Indeed, the random value which blinds P is itself doubled at each execution. The attack goes as follows : a point P is first sent as the first request. Then the card executes its routine with the value $P + R$. The adversary then requests the computation with the point $2P$. With probability $\frac{1}{2}$, the card will use the point $2P + 2R = 2(P + R)$. So the attacker is then able to compare the two measurements and to recover the secret scalar. If the noise is too important, the adversary can use a statistical approach. He can choose a random point Q, send Q and $2Q$ to the card and make the difference between the first curve and the second one shifted by one step to the right. By summing the square of those differences, we can recover all the bits of the secret scalar. Indeed, when the bit at step i is equal to 0, half of the differences at step $i + 1$ are null. So the curve representing the sum of the differences will be flatter at positions corresponding to a zero than at positions corresponding to a one.

H.6 Conclusion

A new powerful attack on scalar multiplication and modular exponentiation has been presented which takes advantage of some implementation choices that were not considered as a security concern up to now. As regards to this attack, it appears that the "bad" choice was the most commonly used, due to efficiency criteria. This vulnerability considerably weakens usual countermeasures used to defeat power attacks.

Since no attack as efficient as the doubling attack is known on the upward double-and-add algorithm (from the least to the most significant bit), we recommend to use this routine to compute scalar multiplication combined with the appropriate countermeasures.

It is an open problem to study whether our attack and Goubin's attack can be combined in order to defeat the combination of Coron countermeasures.

H.7 Appendix 1 : Statistical approach of noise reduction

Let c denotes the number of cycles of a point doubling operation. More precisely, we only consider the cycles that are data dependant, i.e., for which the power consumption differs when operands are changed. The power consumption (without noise) of the computation with A and B at cycle i are respectively named $C_A(i)$ and $C_B(i)$. The noise N can be modeled by random independent variables $N_A(i)$, $N_B(i)$ with mean μ and variance σ. The power consumption observed on cycle i are then equal to $C_A(i)+N_A(i)$ and $C_B(i)+N_B(i)$. The indicator I is defined as follow :

$$\begin{aligned} I &= \frac{1}{c}\sum_{i=1}^{c}(C_A(i)+N_A(i)-C_B(i)-N_B(i))^2 \\ &= \frac{1}{c}\sum_{i=1}^{c}(C_A(i)-C_B(i))^2+\frac{1}{c}\sum_{i=1}^{c}(N_A(i)-N_B(i))^2 \\ &\quad +\frac{2}{c}\sum_{i=1}^{c}(N_A(i)-N_B(i))(C_A(i)-C_B(i)) \end{aligned}$$

If c is large enough, we can evaluate the mean of the indicator when $A=B$ and $A\neq B$. In the first case, $I=\frac{1}{c}\sum_{i=1}^{c}(N_A(i)-N_B(i))^2$, which is a sum of assumed independent variables $Y(i)=(N_A(i)-N_B(i))^2$. The mean of $Y(i)$ is

$$\begin{aligned} E(Y) &= E((N_A(i)-N_B(i))^2) \\ &= Var((N_A(i)-N_B(i))-[E(N_A(i)-N_B(i))]^2 \\ &= Var(N_A(i))+Var(N_B(i))-0=2\sigma^2 \end{aligned}$$

So the mean of the indicator, if $A=B$, is $E(I(A=B))=2\sigma^2$

In the second case ($A\neq B$), assuming that

$$\forall i\leq c, \varepsilon_1\leq |C_A(i)-C_B(i)|\leq \varepsilon_2$$

the mean of the indicator can be bounded as follow

$$2\sigma^2+\varepsilon_1^2\leq E(I(A\neq B))\leq 2\sigma^2+\varepsilon_2^2$$

With this bound in mind, it appears that the indicator can be used to distinguish if the manipulated data are equal or not. The confidence on this indicator relies on its variance and the number of clock cycles c to perform the operation.

Annexe I

Attacking Unbalanced RSA-CRT Using SPA

CHES 2003
[121] avec Gwenaëlle Martinet et Guillaume Poupard

Abstract : *Efficient implementations of RSA on computationally limited devices, such as smartcards, often use the CRT technique in combination with Garner's algorithm in order to make the computation of modular exponentiation as fast as possible. Novak has proposed at PKC 2001 to use some information that may be obtained by simple power analysis on the execution of Garner's algorithm to recover the factorization of the RSA modulus. The drawback of this approach is that it requires chosen messages ; in the context of RSA decryption it can be realistic but if we consider RSA signature, standardized padding schemes make adaptive choice of message representative impossible.*
In this paper, we use the same basic idea than Novak but we focus on the use of known messages. Consequently, our attack applies to RSA signature scheme, whatever the padding may be. However, our new technique based on SPA and lattice reduction, requires a small difference, say 10 bits, between the bit lengths of modulus prime factors.

I.1 Introduction

Since the introduction in 1996 of the timing attacks by Kocher [195], many papers have considered various side channel attacks and the potential countermeasures. Side channels attacks allow to extract some information on the manipulated data which can be used to recover secret data. This general kind of attacks can be divided into several different techniques : Simple Power analysis (SPA) and Differential Power Analysis (DPA), introduced in 1999 by Kocher, Jaffe and Jun [196], timings attacks [195], etc... Lots of countermeasures have been proposed against such attacks but addressing all weaknesses when implementing an algorithm is a hard task. Power attacks are very difficult to prevent, and thus, most of the time, countermeasures do not suffice to thwart all of them.

In the public key setting, many papers have focused on the security of cryptosystems against such side channel attacks [195, 261, 228]. In particular, the RSA signature and encryption schemes have been mainly studied. To sign a message M with RSA, M is first transformed using appropriate padding scheme and hash function into a representative

$m \in \mathbb{Z}_n$, where N is the product of two primes p and q. Then $m^d \bmod N$ is computed, where d is the secret key of the signer. This yields the signature for the message M.

Side channel attacks on RSA extract information from the exponentiation step. For example, by precisely measuring the time the cryptographic device takes to perform the RSA signature, an attacker can recover the secret key, as shown by Kocher in [195]. This timing attack can be mounted with a simple implementation of RSA using the repeated square and multiply algorithm. The attack recovers the secret exponent d, one bit at a time. Indeed, for each non zero bit on the secret exponent d, an additional multiplication is performed. Analyzing differences between running time for various input values reveals the secret key. Another classical way to attack basic RSA implementation is to use SPA technique that consists in measuring the power consumption during exponentiation [89]. Since power consumption also allows to determine if the additional multiplication is done, all bits of the secret exponent can be recovered by monitoring only one exponentiation.

Optimized implementations are also subject to attacks. The Chinese Remainder Theorem is a well known technique to optimize RSA exponentiation. In a CRT implementation, the signer first computes separately the signature modulo each prime factors p and q. He then uses the Chinese Remainder Theorem to compute the signature $S \bmod N$. Since the size of p and q is about half the size of N, CRT exponentiation is about four times faster than direct exponentiation. The first attack on RSA-CRT has been presented in 1997 by Boneh, DeMillo and Lipton [45]. It is based on fault injection during computation. By using a valid signature for a message and a faulty one, the modulus N can be efficiently factored. A timing attacks against RSA with the Chinese Remainder Theorem (CRT) is also possible [261], when the Montgomery algorithm is used for squaring and multiplication operations.

Recently, Novak [228] has described an adaptive chosen message attack against smart cards implementations of RSA decryption when the Garner's algorithm implements the CRT. This attack is based on a simple power analysis (SPA). The power consumption of the card leaks information on the secret manipulated data and the cryptanalyst goal is to relate such information to the bits of the secret key. Although this attack can be mounted against RSA decryption scheme, it is not realistic in practice against the RSA signature scheme. Indeed, a padding scheme is used in practical implementations and then chosen inputs attacks cannot be made.

In this paper, we show how to extend this attack to the RSA signature scheme that uses any encoding scheme such as PKCS#1 [198]. In particular we show that with a simple power analysis the RSA factors p and q can be recovered by performing $60 \times 2^\ell$ signatures on average, for a modulus $N = pq$ where p and q are such that $q < p/2^\ell$, for ℓ larger than an explicit bound we precise in this paper. These signatures can be computed on any messages, not necessarily chosen by the adversary.

In the next section, we briefly describe the RSA-CRT signature scheme implemented with the Garner's algorithm. Then, we develop a new technique to factor N when having access to a set of special form integers modulo N. In the last part of this paper we precisely describe the attack and how to collect such special form RSA signature. Finally we give practical results on experiments of our attack.

I.2 RSA signature scheme

Let $N = pq$ an n-bit RSA modulus. The public key of the signer is denoted by (N, e) and the private key by (p, q, d), where e and d are such that $e \cdot d = 1 \bmod (p-1)(q-1)$. Let M be a message to sign. A signature for M is $S = m^d \bmod N$, where m is deduced

from M by an encoding scheme, randomized or not, such as PKCS#1 for example [198]. To check if a signature S is valid for M, a verifier simply computes m and checks if the equality $m = S^e \bmod N$ holds. Note that the encoding step is mainly used in practice to avoid some basic attacks on RSA. The requirement on the encoding scheme is that the outputs are uniformly distributed in $\mathbb{Z}_n$.

Smart cards implementations of RSA frequently use the Chinese Remainder Theorem to speed up the computation of $S = m^d \bmod N$. The Garner's algorithm is an efficient method to determine the signature S from $s_p = S \bmod p$ and $s_q = S \bmod q$. This algorithm does not require any reduction modulo N but uses instead reductions modulo the factors p and q. It is thus more efficient than the classical implementation of the CRT. A detailed description of this algorithm can be found in [215] and in [192]. In figure 19, we describe the RSA signature generation using the Garner's algorithm.

Algorithme 19 The RSA-CRT signature generation with Garner's algorithm

Require: 5 integers $M, p, q, d, N \geq 0$ with $p > q$, precomputed values : $d_p = d \bmod p - 1$, $d_q = d \bmod q - 1$ and $u = q^{-1} \bmod p$.
Ensure: Signature S of M : $M^d \bmod N$.

1: Encode the message M in $m \in \mathbb{Z}_n$ (with PKCS#1)
2: Compute $s_p = m^{d_p} \bmod p$
3: Compute $s_q = m^{d_q} \bmod q$
4: Set $t = s_p - s_q$
5: If $t < 0$ then $t \leftarrow t + p$
6: Compute $S = s_q + ((t \cdot u) \bmod p) \cdot q$
7: Return S as a signature for the message M
8: **return** M

Step 5 of this algorithm needs some explanation : we first remark that the value t computed at the previous step may be negative, since, as we assume $q < p$, it lies in the range $[-q, p]$. However, the modular multiplication $tu \bmod p$ has to be performed in the next step. Since inputs for modular multiplications have to be already reduced, if t is negative, p should be added so that $t > 0$. Finally, step 5 consists in computing $t \bmod p$.

In [228], Novak has described a method to factor an RSA modulus when the Garner's algorithm is used for CRT. This attack applies to the RSA encryption schemes. It is based on the observation that, for a message m encrypted into c, if $m_p = c^{d_p} \bmod p$ is smaller than $m_q = c^{d_q} \bmod q$, step 5 of the decryption algorithm (similar to 5 of the signature generation in 19) is performed. Otherwise, no addition is made in this step. The analysis of the power trace gives the information on the execution of such a conditional step. In other words, using a SPA analysis, an adversary is able to detect if the addition $t \leftarrow t + p$ is performed, and then to deduce if $m_p < m_q$. This information allows a binary search to recover the factor p : an attacker searches for a plaintext m such that $m \bmod p < m \bmod q$ and $(m-1) \bmod p \geq (m-1) \bmod q$. Such a plaintext can be efficiently found with a binary search combined with a simple power analysis. Once m is found, Novak has remarked that m is in fact a multiple of the factor p that can then be deduced as the GCD of m and the modulus N. However, this attack is only possible in a chosen-plaintext scenario. Thus it cannot be made in practical implementations of the RSA signature scheme due to the encoding step. Indeed, an adversary is still supposed to choose the message M to sign but does not have enough control over the encoding m of M, particularly when randomization techniques are used.

In the following we show how to recover the factor q using this leaked information even if a padding scheme is used. However, we require the prime factors of the modulus to be slightly unbalanced.

I.3 Lattice based techniques

In this part, we describe some basic facts on lattice and we present a new method to factor an RSA modulus based on the lattice reduction algorithm.

I.3.1 Preliminaries on lattices

We denote by $\|\mathbf{x}\|$ the Euclidean norm of the vector $\mathbf{x} = (x_1, \ldots, x_{d+1})$, defined by $\|\mathbf{x}\| = \sqrt{\sum_{i=1}^{d+1} x_i^2}$. Let $\mathbf{v_1}, \ldots, \mathbf{v_d}$, be d linearly independent vectors such that for $1 \leq i \leq d$, $\mathbf{v_i} \in \mathbb{Z}^{d+1}$. We denote by L, the lattice spanned by the matrix V whose rows are $\mathbf{v_1}, \ldots, \mathbf{v_d}$. L is the set of all integer linear combinations of $\mathbf{v_1}, \ldots, \mathbf{v_d}$:

$$L = \left\{ \sum_{i=1}^{d} c_i \mathbf{v_i},\ c_i \in \mathbb{Z} \right\}$$

Geometrically, $\det(L)$ is the volume of the parallelepiped spanned by $\mathbf{v_1}, \ldots, \mathbf{v_d}$. The Hadamard's inequality says that $\det(L) \leq \|\mathbf{v_1}\| \times \ldots \times \|\mathbf{v_d}\|$.

Given $\langle \mathbf{v_1}, \ldots, \mathbf{v_d} \rangle$ the LLL algorithm [8] will produce a so called "reduced" basis $\langle \mathbf{b_1}, \ldots, \mathbf{b_d} \rangle$ of L such that

$$\|\mathbf{b_1}\| \leq 2^{(d-1)/2} \det(L)^{1/d} \tag{I.1}$$

in time $O(d^4 \log(M))$ where $M = \max_{1 \leq i \leq d} \|\mathbf{v_i}\|$. Consequently, given a basis of a lattice, the LLL algorithm finds a short vector $\mathbf{b_1}$ of L satisfying equation (I.1). Moreover, we assume in the following that the new basis vectors are of the same length and also have all their coordinates of approximatively the same length. Indeed, a basis for a random lattice can be reduced into an almost orthonormal basis. Therefore, $\|b_i\| \approx \|b_1\|$ for $1 \leq i \leq d$, and so $\|b_i\|^d \approx \det(L)$.

I.3.2 Factoring using LLL

In the following we describe a new method, based on lattice reduction, to factor a modulus N given some special form integers s_i in $\mathbb{Z}_n$.

Let $N = p \times q$ be an RSA modulus such that p and q are two prime integers. Let s_1, s_2, ..., s_d be d integers from $\mathbb{Z}_n$. For each s_i, we consider its euclidian division by p

$$\forall i \in [1, d] \quad s_i = r_i + u_i \times p \quad \text{with} \quad r_i \in \mathbb{Z}_p \quad \text{and} \quad u_i \in \mathbb{Z}_q$$

Let us assume that, instead of being distributed all over the set $\mathbb{Z}_p$, the r_is are smaller than a bound $A < p/2$.

We further consider the lattice L spanned by the $d+1$ rows of the following matrix

$$\begin{pmatrix} N & 0 & \ldots & \ldots & 0 \\ 0 & N & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \ldots & 0 & N & 0 \\ -s_1 & -s_2 & \ldots & -s_d & A \end{pmatrix}$$

Théorème I.1. *Assuming the LLL algorithm is able to compute the shortest vector of a lattice, the reduction of lattice L computes the factorization of modulus N with probability* $> 1 - \varepsilon_0$ *if the difference of bit-length of p and A is such that*

$$\log p - \log A > \max\left(\frac{\log q - \log \varepsilon_0 - 1.105}{d} + 2.047, \log\left(2\sqrt{d+1}\right)\right)$$

As an example, for a 512-bit prime factor q and a probability of success of the algorithm $> 1 - 2^{-10}$, we obtain a minimum $\log p - \log A \approx 10.4$ for $d = 60$. Note that this minimum does not strongly depend on the size of q and is about 5 bits for any cryptographic size of this factor.

Sketch of proof. [A complete proof is proposed in appendix I.6]
By definition of the lattice L, any vector of L is an integer combination of the rows of the matrix. In other words, we may define the lattice in the following way :

$$L = \left\{(c_1 N - cs_1, c_2 N - cs_2, \ldots, c_d N - cs_d, Ac)\ ;\ (c_1, c_2, \ldots, c_d, c) \in \mathbb{Z}^{d+1}\right\}$$

For a fixed choice of the integer coefficients $(c_1, c_2, \ldots, c_d, c)$, we note

$$b(c_1, c_2, \ldots, c_d, c) = (c_1 N - cs_1, c_2 N - cs_2, \ldots, c_d N - cs_d, Ac) \in L$$

In other words, the lattice L is the set of all the vectors $b(c_1, c_2, \ldots, c_d, c)$ for $(c_1, c_2, \ldots, c_d, c) \in \mathbb{Z}^{d+1}$.

A special vector of the lattice, strongly related with the q prime factor of N, is "abnormally" short. The consequence is that we can expect the LLL lattice reduction algorithm to compute this short vector.

This special vector is $b^* = b(u_1, u_2, \ldots, u_d, q)$, where u_i is defined by division of the s_i by p, and q is the other factor of the modulus $N = p \times q$. Note that the knowledge of b^* immediately reveals q since its last coordinate is Aq and A is known. The size of b^*, i.e. its euclidian norm, can be easily estimated and we obtain $||b^*|| < \sqrt{d+1}Aq$.

Then, in order to prove that the vector b^* is the shortest one, we study $||b(c_1, c_2, \ldots, c_n, c)||$ and we prove that, if $c \neq q$, those vectors are larger that b^*, whatever the c_is may be, for all the s_i but only a very small fraction. A precise analysis, described in appendix I.6, leads to the result of theorem I.1.

Therefore the knowledge of d values s in $\mathbb{Z}_n$ such that $|s \bmod p| < A$ allows us to factor N. In the following, we describe how SPA can be used to find such d values in the context of RSA signature generation, with "slightly" unbalanced modulus.

I.4 Application to RSA-CRT signature scheme

In this section, we use the results presented above to extend the chosen ciphertext attack described by Novak in [228]. In particular, we show that if the factors p and q of the modulus are such that $|p| - |q| > \ell$, for a given bound ℓ , then they can be recovered with a known message attack combined with a simple power analysis.

In the following we suppose that a SPA attack allows us to detect if the addition of step 5 is performed during a signature generation. Such an assumption is realistic in practice if no countermeasure is implemented, and a detailed way to extract this information can be found in [228] and in [195].

Attack. In the previous section we have shown how the prime factors p and q of a modulus N can be recovered, given a set of integers s in $\mathbb{Z}_n$ such that $s \bmod p$ is less than a given bound. We apply this result by using a simple power analysis on the RSA signature scheme in order to find these integers.

We assume that the prime factors p and q are such that $|p| - |q| > \ell$, for ℓ a small integer. Such an assumption is realistic in many actual implementations since, in many descriptions of the RSA algorithm, we can find that p and q have to be of "roughly" the same length, or "about the same bit-length". Here, we consider that p and q have a very small bit-length difference, about 10 bits for an 1024-bit modulus. This does not constitute a contradiction with the usual description of RSA, that can be interpreted in many different ways.

Let S be a signature for a random message M, computed by using the algorithm described in figure 19. We suppose that the step 5 of the algorithm has been performed to generate the signature S. Otherwise, we choose another random message until this step is executed. Since the optional addition has been made, we know that $s_p - s_q < 0$. Thus, since we assume that $q < p$, we simply have that $s_p < s_q < q < p$. By definition of $s_p = S \bmod p$, S can always be written as $S = s_p + u \times p$ for an integer $u < q$. Consequently S is a candidate input to the factoring algorithm, given in section I.3.2, for the upperbound $A = q$ on $s \bmod p$. The problem here is that clearly, this bound is not known to the attacker. Thus, the last entry A of the matrix cannot be explicitely given. However, A is an upperbound on the $s_i \bmod p$ where s_i is the ith input of the last row. Thus choosing an integer $A > q$ is a correct choice and we choose in practice A to be the larger integer such that $|A| = |q|$.

To run the lattice reduction described in the previous section, we have to find d signatures s_i such that $s_i \bmod p$ is less than the bound A we choose. We thus query the signature of messages and we perform a SPA attack on each generation. Each signature verifying $s_p < s_q$ is kept as input to the matrix. We query the signing card until d valid candidate signatures have been found.

The number d of required signatures has to be sufficiently large, according to the bit-length difference ℓ between the factors p and q, and according to the modulus length. To estimate the average number of queries made to the signing card, we compute the probability that $s_p = m^{d_p} \bmod p$ is less than $s_q = m^{d_q} \bmod q$, for $q < p/2^\ell$, and for random integers $s \in \mathbb{Z}_n$. We suppose that the values s are uniformly and independently distributed in $\mathbb{Z}_n$. This is verified in practice when an appropriate hash function, such as SHA-1, is used. In this case we assume that the output m of the encoding scheme is uniformly distributed in $\mathbb{Z}_n$ so that $s = m^d \bmod N$ is uniformly distributed. Let s_p and s_q the values computed during the signature generation in steps 2 and 3 respectively. We have :

$$\begin{aligned}
\Pr s_p < s_q &= \sum_{B=0}^{q-1} \Pr s_p < B | s_q = B \cdot \Pr s_q = B \\
&= \sum_{B=0}^{q-1} \frac{B}{p} \cdot \frac{1}{q} = \frac{q(q-1)}{2pq} \\
&< \frac{1}{2^{\ell+1}}
\end{aligned}$$

where the last inequality comes from the fact that $q < p/2^\ell$.

Thus in a set of 2^ℓ signatures, there is a probability of one half that at least one of them is such that $s_p < s_q$. Detecting such a signature is possible with a SPA attack during

the signature generation : when the step 5 is performed, we know that the signature S_i is a good candidate input for our algorithm, if we write it as $S_i = (S_i \bmod p) + u_i p$ where $S_i \bmod p < p/2^\ell$. Otherwise, we query another signature until we find a good candidate. On average, 2^ℓ trials are needed. To have d such signatures, a set of $d \cdot 2^\ell$ signatures is required. The algorithm described in section I.3.2 to factor a modulus N can then be used with the candidate signatures as inputs. Following the analysis made above, the number d of required signatures must be such that $\ell \geq \frac{\log q - 10}{d} + 2$ so that the algorithm successfully ends with probability greater than $1 - 2^{-10}$. However, we assume here that $q < p/2^\ell$, and thus, $\log q = \frac{\log N - \ell}{2}$. Thus, we have :

$$\begin{aligned} \ell &\geq \frac{\log q - 10}{d} + 2 \\ &\geq \frac{\frac{\log N - \ell}{2} - 10}{d} + 2 \\ &\geq \frac{\log N - 20 + 4d}{2d + 1} \end{aligned}$$

Thus, if p and q are such that $q < p/2^\ell$ for ℓ greater than $\frac{\log N - 20 + 4d}{2d+1}$, we can factor N from $d \times 2^\ell$ signatures on average and with probability at least $1 - 2^{-10}$.

Experimental results. In practice, the lattice dimension depends on the value ℓ, and on the number of available signatures. However, if the lattice dimension is too large, then the LLL algorithm fails. Particularly, under a reasonable time, it is not possible to run the LLL algorithm on a 100×100 dimensional matrix where the entries are 1024-bit numbers.

For each modulus length, we give in figure I.1 the integer ℓ such that $q < p/2^\ell$, the dimension $d + 1$ of the lattice, the average number of required signatures and the time needed to recover p and q. The number of required signatures, equal to $d \times 2^{|p|-|q|}$, is upperbounded by $2^{6+|p|-|q|}$ since we always have $d < 2^6$. The tests have been run on an Intel Pentium IV, XEON 1.5 GHz, with the Victor Shoup's library NTL ([271]).

From this previous table, we show that the LLL algorithm works better than the theoretical results indicated in section I.3.2. For 1024-bit modulus, the expected result gives $\ell > 10$. These values are realistic since a difference of 10 bits between p and q for a 1024-bit modulus means that p is a 517-bit prime and q is a 507-bit prime number. This distance between p and q is not ruled out by the specifications of the key generation of the RSA algorithm. It is worth noticing that this attack can be extended to more "secure" moduli, say 2048 bits long. Moreover, this attack can be mounted in practice since the number of required signatures is not large, one million if $n = 1536$, and since it is only a known message attack.

I.5 Conclusion

Implementations of the Garner's algorithm has to be carefully studied so that SPA attacks should not be possible on the card. In particular, constant running time, independent of the secret key bits, or blinding techniques are essential.

modulus length in bits	$\lvert p\rvert - \lvert q\rvert$	lattice dimension $d+1$	average number of required signatures	time to factor
512	8	41	2^{13}	30 s
	7	51	2^{13}	2 min
	6	61	2^{12}	14 min
768	10	43	2^{15}	50 min
	9	51	2^{15}	2 min
	8	61	2^{14}	15 min
1024	12	46	2^{17}	2 min
	11	56	2^{17}	6 min
	10	61	2^{16}	16 min
1536	16	53	2^{22}	7 min
	15	56	2^{21}	10 min
	14	61	2^{20}	32 min
2048	20	53	2^{26}	11 min
	19	56	2^{25}	20 min
	18	61	2^{24}	32 min

FIGURE I.1: Experimental results on the RSA-CRT with unbalanced modulus.

I.6 Appendix : Proof of theorem I.1

By definition of the lattice L, any vector of L is an integer combination of the rows of the matrix. In other words, we may define the lattice in the following way :

$$L = \left\{ (c_1 N - cs_1, c_2 N - cs_2, \ldots, c_d N - cs_d, Ac) \; ; \; (c_1, c_2, \ldots, c_d, c) \in \mathbb{Z}^{d+1} \right\}$$

For a fixed choice of the integer coefficients $(c_1, c_2, \ldots, c_d, c)$, we note

$$b(c_1, c_2, \ldots, c_d, c) = (c_1 N - cs_1, c_2 N - cs_2, \ldots, c_d N - cs_d, Ac) \in L$$

In other words, the lattice L is the set of all the vectors $b(c_1, c_2, \ldots, c_d, c)$ for $(c_1, c_2, \ldots, c_d, c) \in \mathbb{Z}^{d+1}$.

We now show that a special vector of the lattice, strongly related with the q prime factor of N, is "abnormally" short. The consequence is that we can expect the LLL lattice reduction algorithm to compute this short vector.

This special vector is $b^* = b(u_1, u_2, \ldots, u_d, q)$, where u_i is defined by division of the s_i by p, and q is the other factor of the modulus $N = p \times q$. Note that the knowledge of b^* immediately reveals q since its last coordinate is Aq and A is known. Let us evaluate the size of b^*, i.e. its euclidian norm

$$\begin{aligned} ||b^*||^2 &= ||b(u_1, u_2, \ldots, u_d, q)||^2 \\ &= ||(u_1 N - qs_1, u_2 N - qs_2, \ldots, u_d N - qs_d, Aq)||^2 \\ &= \sum_{i=1}^{d} (u_i N - q(r_i + u_i p))^2 + A^2 q^2 \\ &= \sum_{i=1}^{d} q^2 r_i^2 + A^2 q^2 \end{aligned}$$

Since $0 \leq r_i < A$, we immediately obtain $||b^*||^2 < dq^2A^2 + A^2q^2$, so

$$||b^*|| = ||b(u_1, u_2, \ldots, u_d, q)|| < \sqrt{d+1}Aq$$

In order to prove that the vector b^* is abnormally short, we now show that the other elements of the lattice L are, with overwhelming probability, larger. With this aim in view, we define, for any integer c, the function

$$\mathcal{F}(c) = \min_{(c_1,c_2,\ldots,c_d)\in\mathbb{Z}^d} ||b(c_1, c_2, \ldots, c_d, c)||$$

i.e., $\mathcal{F}(c)$ is the size of the shortest vector in L whose last coordinate is equal to Ac. From the definition of $\mathcal{F}(c)$, we can derive the following expression

$$\mathcal{F}(c) = \min_{(c_1,c_2,\ldots,c_d)\in\mathbb{Z}^d} \sqrt{\sum_{i=1}^{d}(c_iN - cs_i)^2 + A^2c^2}$$

Then, it is easy, for a fixed c, to find the c_is that reach the minimum since $\mathcal{F}(c)$ is the minimum of a sum of independent squares. As a consequence, the minimum is reached when each term is as small as possible. This means that c_i is the nearest integer of cs_i/N ; we note $\left\lfloor \frac{cs_i}{N} \right\rceil$ this integer. We finally obtain

$$\mathcal{F}(c) = \sqrt{\sum_{i=1}^{d}\left(\left\lfloor \frac{cs_i}{N} \right\rceil N - cs_i\right)^2 + A^2c^2}$$

We can now notice that, by definition of $\mathcal{F}(c)$ and $b^* = b(u_1, u_2, \ldots, u_d, q)$,

$$\mathcal{F}(q) \leq ||b^*|| < \sqrt{d+1}Aq$$

In fact, b^* is exactly the smallest vector with last coordinate equal to Aq because

$$\begin{aligned}
\mathcal{F}(q)^2 &= \sum_{i=1}^{d}\left(\left\lfloor \frac{qs_i}{N} \right\rceil N - qs_i\right)^2 + A^2q^2 \\
&= q^2 \times \left(\sum_{i=1}^{d}\left(\left\lfloor \frac{s_i}{p} \right\rceil p - s_i\right)^2 + A^2\right) \\
&= q^2 \times \left(\sum_{i=1}^{d}\left(\left\lfloor \frac{r_i + u_i \times p}{p} \right\rceil p - r_i - u_i \times p\right)^2 + A^2\right) \\
&= q^2 \times \left(\sum_{i=1}^{d}\left(\left\lfloor \frac{r_i}{p} \right\rceil p - r_i\right)^2 + A^2\right) \\
&= q^2 \times \left(\sum_{i=1}^{d} r_i^2 + A^2\right)
\end{aligned}$$

since, for $0 \leq r_i < A < \frac{p}{2}$, $\left\lfloor \frac{r_i}{p} \right\rceil = 0$. This finally proves that $\mathcal{F}(q) = ||b^*||$.

Then, we can further notice that if $c > \sqrt{d+1} \times q$, $\mathcal{F}(c)$ is obviously greater than Ac so $\mathcal{F}(c) > \sqrt{d+1} \times q \times A > \mathcal{F}(q)$.

We finally need to evaluate, for a fixed $c \neq q$, the probability that $\mathcal{F}(c) < \mathcal{F}(q)$ where the probabilities are computed when the s_i are uniformally distributed in

$$\mathcal{S} = \{r + u \times p; 0 \leq r < A, 0 \leq u < q\} \subset \mathbb{Z}_N$$

If this probability is negligible, we can conclude that b^* is, with overwhelming probability, the shortest vector of the lattice.

We first notice that the distribution of the $\left\lfloor \frac{c\times s_i}{N} \right\rceil \times N - c \times s_i$, when s_i is uniformally distributed in $\mathbb{Z}_N$, is uniform between $\frac{-(N-1)}{2}$ and $\frac{N-1}{2}$. This is an obvious consequence of the fact that $\left\lfloor \frac{c\times s_i}{N} \right\rceil - \frac{c\times s_i}{N} \in [-1/2; 1/2]$ and that if $\left\lfloor \frac{c\times s_i}{N} \right\rceil \times N - c \times s_i = \alpha$ for an integer α, we obtain by modular reduction that $-c \times s_i = \alpha \bmod N$ and consequently that $s_i = -\alpha \times c^{-1} \bmod N$ if c is prime with N.

If we restrict the possible values of s_i to the set $\mathcal{S}$, the previous result implies

$$\mathcal{D}_c = \left\{ \left\lfloor \frac{cs_i}{N} \right\rceil \times N - cs_i \ ; \ s_i = r_i + u_i \times p \in \mathcal{S} \right\} \subset \left[\frac{-(N-1)}{2}, \frac{N-1}{2} \right]$$

We can further write

$$\begin{aligned}
\mathcal{D}_c &= \left\{ \left\lfloor \frac{cs_i}{N} \right\rceil \times N - cs_i \ ; \ s_i = r_i + u_i p \ , \ 0 \le r_i < A \ , \ 0 \le u_i < q \right\} \\
&= \left\{ \left\lfloor \frac{cr_i}{N} + \frac{cu_i}{q} \right\rceil \times N - cr_i - cu_i p \ ; \ 0 \le r_i < A \ , \ 0 \le u_i < q \right\} \\
&= \left\{ \left\lfloor \frac{cr_i}{N} + \frac{cu_i \bmod q}{q} \right\rceil \times N - cr_i - (cu_i \bmod q) \times p \ ; \ 0 \le r_i < A \ , \ u_i \in \mathbb{Z}_q \right\} \\
&= \left\{ \left\lfloor \frac{cr_i}{N} + \frac{v_i}{q} \right\rceil \times N - cr_i - v_i p \ ; \ 0 \le r_i < A \ , \ 0 \le v_i < q \right\}
\end{aligned}$$

Then, for any $\alpha \in [-(N-1)/2, (N-1)/2]$, if $\left\lfloor \frac{c\times r_i}{N} + \frac{v_i}{q} \right\rceil \times N - c \times r_i - v_i \times p = \alpha$ we obtain that $\alpha = -cr_i - v_i p \bmod N$. Since v_i is uniformally distributed in $\mathbb{Z}_q$ and $\alpha + p = -cr_i - (v_i - 1 \bmod q) \times p \bmod N$, we conclude that, if α is an element of $\mathcal{D}_c$, $\alpha + p$ (or $\alpha + p - N$ if $\alpha + p > N/2$) is also an element of $\mathcal{D}_c$. So, if $\gcd(c, N) = 1$, the set $\mathcal{D}_c$ is a subset of $\left[\frac{-(N-1)}{2}, \frac{N-1}{2} \right]$ that is invariant by (circular) translation of length p.

In other words, this formalizes the idea that, if c is prime with N, the elements of $\mathcal{D}_c$ are well distributed in $\left[\frac{-(N-1)}{2}, \frac{N-1}{2} \right]$. As a toy example, figure I.2 represents the elements of $\mathcal{D}_3 \times \mathcal{D}_3$ for $N = 7 \times 11$ and $A = 5$.

Always with the aim of computing the probability for $\mathcal{F}(c)$ to be less that $\mathcal{F}(q)$, if $\gcd(c, N) = 1$, we can state that

$$\begin{aligned}
&\Pr\left(s_1, s_2, \ldots, s_d\right) \in \mathcal{S}^d \mathcal{F}(c) < \mathcal{F}(q) < \Pr\left(s_1, s_2, \ldots, s_d\right) \in \mathcal{S}^d \mathcal{F}(c) < \sqrt{d+1} A q \\
&< \Pr\left(s_1, s_2, \ldots, s_d\right) \in \mathcal{S}^d \sum_{i=1}^{d} \left(\left\lfloor \frac{cs_i}{N} \right\rceil N - cs_i \right)^2 < (d+1) A^2 q^2 - A^2 c^2 \\
&< \Pr\left(s_1, s_2, \ldots, s_d\right) \in \mathcal{S}^d \sum_{i=1}^{d} \left(\left\lfloor \frac{cs_i}{N} \right\rceil N - cs_i \right)^2 < (d+1) A^2 q^2
\end{aligned}$$

Let $\Delta = \sqrt{d+1} A / p$; we need to evaluate the probability

$$\Pr\left(s_1, s_2, \ldots, s_d\right) \in \mathcal{S}^d \sum_{i=1}^{d} \left(\left\lfloor \frac{cs_i}{N} \right\rceil N - cs_i \right)^2 < N^2 \Delta^2$$

From the previous result on the distribution of $X_i = \left\lfloor \frac{cs_i}{N} \right\rceil - \frac{cs_i}{N}$ in $\left[-\frac{1}{2}, \frac{1}{2} \right]$, this probability can be approximated, for large N, by

$$\Pr\left(X_1, X_2, \ldots, X_d\right) \in_u \left[-\frac{1}{2}, \frac{1}{2} \right] \sum_{i=1}^{d} X_i^2 < \Delta^2$$

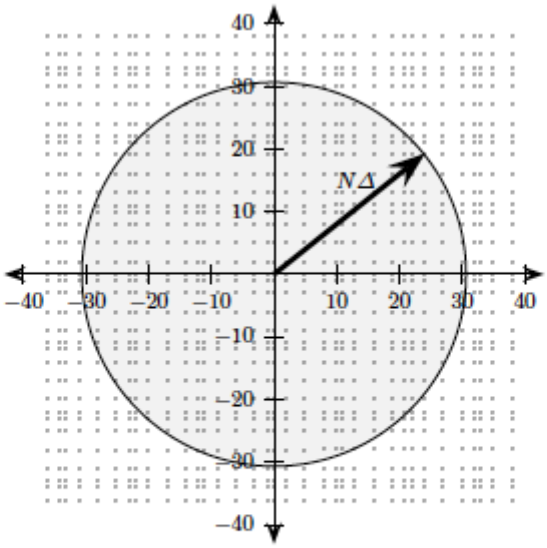

Fig. 3. Graphical representation of the pairs (s_1, s_2) for the parameters $d = 2$, $p = 11$, $q = 7$, $N = 77$ and $A = 5$. The disk of radius $N\Delta$ covers a ratio of those points that is illustrated in figure 4

FIGURE I.2: Graphical representation of the pairs (s_1, s_2) for the parameters $d = 2$, $p = 11$, $q = 7$, $N = 77$ and $A = 5$. The disk of radius $N\Delta$ covers a ratio of those points that is illustrated in figure I.3.

where the X_is are independent and uniformally distributed over $\left[-\frac{1}{2}, \frac{1}{2}\right]$. If $0 \leq \Delta < \frac{1}{2}$, this probability is equal to the volume of the d-dimensional ball of radius Δ. Figure I.3 provides a graphical illustration of this fact, using the toy example of figure I.2. If d is even, this volume is equal to $\pi^{d/2}\Delta^d/(d/2)!$.

Note that this approximation of the number of points in the ball of radius $N\Delta$ (see figure I.2) using the volume of this ball is very good for values of c that are relatively prime with q, even if the repartition of the points is not perfectly uniform. This is mainly due to the compensation of local errors of estimation. When $c = q$, such a compensation does not apply and the approximation can no longer be used.

Then, using the well-known Stirling formula, we obtain the upper-bound

$$\Pr\left(s_1, s_2, \ldots, s_d\right) \in \mathcal{S}^d \sum_{i=1}^{d} \left(\left\lfloor \frac{cs_i}{N} \right\rceil N - cs_i\right)^2 < N^2\Delta^2 < \left(\frac{2e\pi\Delta^2}{d}\right)^{\frac{d}{2}} \times \frac{1}{\sqrt{d\pi}}$$

We finally obtain

$$\Pr\left(s_1, s_2, \ldots, s_d\right) \in \mathcal{S}^d \mathcal{F}(c) < \mathcal{F}(q) < \left(\frac{2e\pi\Delta^2}{d}\right)^{\frac{d}{2}} \times \frac{1}{\sqrt{d\pi}}$$

Note that this is true only if $\Delta < \frac{1}{2}$, i.e., if $A < \frac{p}{2\sqrt{d+1}}$. In other words, this means that the difference $-\log(A/p)$ of bit-length of p and A must fulfill the following inequality in order to make the proof correct

$$-\log\left(\frac{A}{p}\right) > \log\left(2\sqrt{d+1}\right)$$

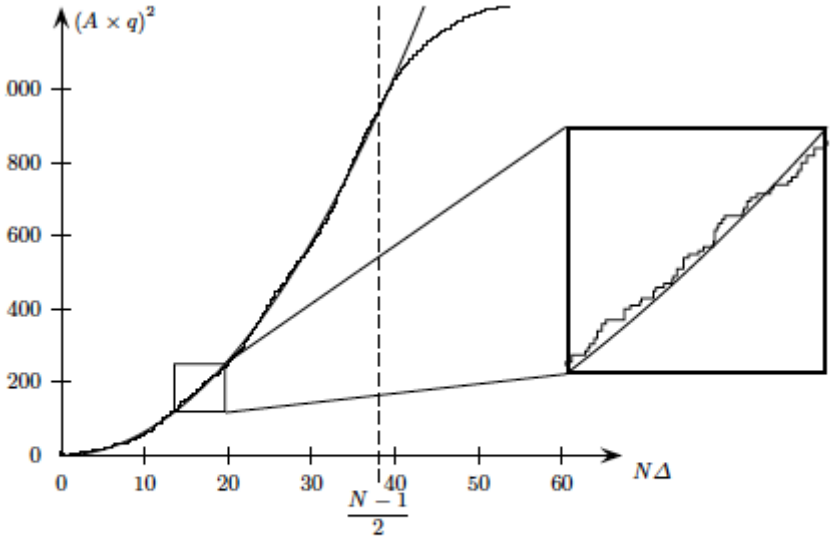

Fig. 4. Ration of points in figure 3 covered by a disk of radius $N\Delta$. The irregular experimental curve represents the number of points covered by the disk and the smooth curve is based on the approximation by the surface of this disk.

FIGURE I.3: Ration of points in figure I.2 covered by a disk of radius $N\Delta$. The irregular experimental curve represents the number of points covered by the disk and the smooth curve is based on the approximation by the surface of this disk.

We finally obtain an upper bound of the probability for $\mathcal{F}(c)$ to be smaller than $\mathcal{F}(q)$

$$\begin{aligned}\Pr\left(s_1, s_2, \ldots, s_d\right) \in \mathcal{S}^d \mathcal{F}(c) < \mathcal{F}(q) &< \left(\frac{2e\pi A^2}{p^2} \times \frac{d+1}{d}\right)^{\frac{d}{2}} \times \frac{1}{\sqrt{d\pi}} \\ &< \sqrt{\frac{e}{d\pi}}\left(\sqrt{2\pi e}\right)^d \left(\frac{A}{p}\right)^d\end{aligned}$$

using the fact that $\left(\frac{d+1}{d}\right)^d < e$.

The last step is to estimate for which values of the parameters we can consider that $\mathcal{F}(c) > \mathcal{F}(q) = ||b^*||$ for any $c \neq q$. Let $P_0 = 1 - \varepsilon_0$ be a lower bound of the probability for b^* to be the smallest vector of the lattice. From the previous results, using an approximative argument of independence of the probabilities for different values of c, we deduce

$$\Pr\left(s_1, s_2, \ldots, s_d\right) \in \mathcal{S}^d \forall c \neq q \; \mathcal{F}(c) > \mathcal{F}(q) > \left(1 - \sqrt{\frac{e}{d\pi}}\left(\sqrt{2\pi e}\right)^d \left(\frac{A}{p}\right)^d\right)^{\sqrt{d+1} \times q}$$

This last expression is greater that P_0 if

$$\varepsilon_0 > \left(\sqrt{d+1} \times q\right) \times \sqrt{\frac{e}{d\pi}}\left(\sqrt{2\pi e}\right)^d \left(\frac{A}{p}\right)^d$$

The inequality can be reworded as follows

$$\begin{aligned}
-\log\left(\frac{A}{p}\right) &> \frac{1}{d}\left(-\log\varepsilon_0 + \frac{1}{2}\log\left(\frac{d+1}{d}\right) + \log q + \frac{1}{2}\log\left(\frac{e}{\pi}\right) + d\log\left(\sqrt{2\pi e}\right)\right) \\
&> \frac{\log q - \log\varepsilon_0 + \frac{1}{2}\log\left(\frac{e}{\pi}\right)}{d} + \log\left(\sqrt{2\pi e}\right) \\
&> \frac{\log q - \log\varepsilon_0 - 1.105}{d} + 2.047
\end{aligned}$$

In other words, this means that, if the difference $-\log(A/p)$ of bit-length of p and A is larger than $\frac{\log q - \log\varepsilon_0 - 1.105}{d} + 2.047$, the algorithm finds q with probability $> 1 - \varepsilon_0$, assuming that LLL return the shortest vector of the lattice.

Annexe J

Power Attack on Small RSA Public Key

CHES 2006

[114] avec Sébastien Kunz-Jacques, Gwenaëlle Martinet, Frédéric Muller et Frédéric Valette

Abstract : *In this paper, we present a new attack on RSA when the public exponent is short, for instance* 3 *or* $2^{16}+1$*, and when the classical exponent randomization is used. This attack works even if blinding is used on the messages.*
From a Simple Power Analysis (SPA) we study the problem of recovering the RSA private key when non consecutive bits of it leak from the implementation. We also show that such information can be gained from sliding window implementations not protected against SPA.

J.1 Introduction

Simple Power Analysis and Differential Power Analysis attacks are among the most efficient and devastating attacks on some RSA-based products. Many countermeasures have been proposed that prevent these attacks by securing the exponentiation algorithm which is usually targeted. This is the basis of a number of academic papers whose results are widely used in practice. However, such countermeasures often lead to a slower implementation and thus another area of research is the speedup of the exponentiation process. As we will show in this article, unfortunate interactions between side-channel countermeasures and optimized exponentiation algorithms may lead to insecure implementations.

The aim of this paper is to present a new attack on RSA in the special case where both a short public exponent and a randomization of the private exponent are used. In such a case, free information on the private exponent can be obtained from the public key and can be used to efficiently recover the whole private key. The attack studies the problem when non-consecutive bits of the private key can be found. It works on sliding window implementations not protected against SPA attack.

Known results on partially known information.

Partial information on the RSA private key allows it to be recovered in some cases. This kind of attacks has experienced a revival since 1998 with the work of Boneh, Durfee and Frankel [47]. In their article, they give some results about the security of RSA schemes when some bits of the private key are exposed. However, the lattice technique used in such cases cannot be applied for non-consecutive bits. None of the previous papers have considered this particular case. Boneh *et al.* have even considered in [47] that *"[the authors] view attacks that require non-consecutive bits of d as artificial"*, showing the lack of interest for this topic at that time.

However, some practical attacks are now very efficient and allow the attacker to recover some bits of the private key, not necessarily consecutive. This is mainly due to the combination of very specialized attacks, based on side channel analysis, and of the various countermeasures based on algorithmic remarks.

Main idea of the attack.

Here, we do not solve the open problem of recovering the whole private key from non-consecutive partial information on it. However, we focus on the special case where several non-consecutive bits of randomized versions of the private key are known.

Efficient countermeasures against SPA attack are often not perfect. It is classical that some information about the secret exponent leaks. For example, sliding window implementations can leak when consecutive bits are equal to zero. By randomly generating bitstring and applying the parsing exponent algorithm of the sliding window algorithm, either Constant Length Non-zero Window (CLNW) or Variable Length Non-zero Window (VLNW), one can observe that the information gained is 40% of the bits. This is not sufficient to recover the entire secret by using previous results such as those of [47].

The first part of the attack is to record some power curves C_i which correspond to the exponentiation of a message with the unknown private key $d_i = d + \lambda_i \times \varphi(N)$ associated to an unknown short value λ_i. Then, using the fact that the public exponent is small, we can consider that the most significant bits of the secret exponent d are known. With this information, we can try all the possible values of λ and check if the most significant bits of the value $\tilde{d} + \lambda \cdot N$, where $\tilde{d}$ equals d on the half bits of high order, are compatible with the partial information that can be recovered from the power curve by SPA. If we have enough information, we can associate a single λ_i to the curve C_i.

The second part of the attack is now to recover the least significant bits of d. Once we have enough curves with the known random value λ_i we can then use partial information on the least significant bits of the randomized exponent on all the curves to retrieve the least significant bits of the secret exponent. The principle is to guess the least significant bits of $\varphi(N)$ and so of d and of the secret exponent d_i. Then, we check if the guess is compatible with the partial information on the curve C_i. If we have enough curves, only one guess will be compatible. We can then guess the next bits and continue until we know enough bits of d.

Our results.

We recall in section J.2 how to get non-consecutive bits of the RSA secret exponent by using side channel attacks. Such leakage depends on the exponentiation algorithm used and on the various countermeasures implemented against side channel attacks.

Then, in section J.3 we formally show how to recover the whole RSA secret key from such information in the case of the public exponent is 3. We extend this attack for $e =$

$2^{16}+1$ in section J.4, and we give practical results in section J.5.

Related Works.

A lot of work has already been done on the particular topic of attacks when countermeasures are implemented. Indeed, a countermeasure may allow or simplify a side channel attack.

Previous works have also been done to study the security of fast exponentiation algorithm. Walter, in [286], describes the Big Mac attack which works on sliding and m-ary window algorithms. He assumes that he can distinguish *squares and multiplies* and *operand of the multiplies*. Here, we only assume that we can distinguish *squares and multiplies* but we do not need to distinguish the different operands of the multiplications.

Walter has also see in [287] that *"in the classical m-ary and sliding windows exponentiation algorithms, the most significant half of the public modulus yields information which can be used to halve the number of key digits which need to be guessed."* Having reduce the key digit by half or a quarter is not sufficient for an 1024-bit value since 256 are missing.

The problem of computing the RSA private exponent from partial information on it has known only a little attention in the literature. In [275], Stinson presents two algorithms to compute discrete logarithms in a prime field, when the Hamming weight of the discrete log is small. As we want to recover d such that $s = f(H(m))^d \bmod N$, where f is the padding function, and we know s and $f(H(m))$, d can be viewed as the discrete log of s in basis $f(H(m))$ in $\mathbb{Z}_N^*$. In appendix J.6, we show that this algorithm can be used to recover d from non-consecutive bits of it if the number of missing bits is relatively small, 128 for instance. However the memory and time complexity of this algorithm is high compared to our algorithm and cannot recover a large number of bits.

J.2 Modular exponentiation and side channel attacks

J.2.1 Classical countermeasures against side channel attacks

To defeat DPA attacks, many protections methods have been suggested in the literature. The most secure and widely used is the exponent randomization [195] as it is very easy to implement and it comes at a reasonable computational cost. The idea of this countermeasure is to use a classical SPA-protected implementation of the exponentiation and to randomize the private exponent at each computation. This randomization is based on the fact that the private exponent d is defined modulo $\varphi(N)$ since for all $M \in \mathbb{Z}_N^\star$ and all $\lambda \in \mathbb{Z}$, $M^{\lambda \times \varphi(N)} = 1 \bmod N$. Figure J.1 describes this randomized exponentiation algorithm.

- `Inputs` : a message M, an exponent d, a modulus N and $\varphi(N)$
- `Output` : $M^d \bmod N$

1. Pick at random $\lambda \in \{0, \ldots, 2^\ell - 1\}$
2. Compute $d' = d + \lambda \cdot \varphi(N)$
3. Return SPA protected exponentiation $M^{d'} \bmod N$

FIGURE J.1: The exponent randomization algorithm

The success of this countermeasure lies in its very good efficiency and the security it offers. Indeed, without randomization an attacker is able to guess the exponent bit per bit and his check would be confirmed with a DPA attack [217]. With the randomization, such a guess cannot be made anymore on the value d' since the attacker does not know the random value used.

J.2.2 Optimized exponentiation algorithms

Timing attacks or SPA attacks are known to be very efficient on RSA-based cryptosystems. In [195], Kocher has shown how to recover the whole private key from the power consumption of a single RSA signature or decryption. If the square-and-multiply algorithm is used for the exponentiation without any countermeasure, various side channel attacks may be used to compromise the private key.

More efficient exponentiation algorithms may be used. Some of them use a parsing of the private exponent into windows of constant or variable length. In that case, side channel attacks cannot recover the whole private exponent anymore. Only some bits of it leak from the implementation, whose distribution depends on technical details of the exact algorithm used. The m-ary or the sliding window techniques are such methods.

The sliding window methods. Such methods have been developed to speed up the exponentiation algorithm by searching in the exponent large windows of bits equal to zero. Contrary to the m-ary algorithm, sliding window methods relax the splitting of the exponent into *sliding windows*. There are two variants known as the Constant Length Non-zero Window (CLNW) technique due to Knuth [192] and the Variable Length Non-zero Window (VLNW) technique due to Bos and Coster in [51]. Both of these techniques try to minimize the number of multiplications in the square-and-multiply algorithm by performing some precomputations. These techniques have been described and analyzed by Koç in [193, 194] and allow 5 to 8% of the multiplications to be avoided compared to the binary exponentiation algorithm.

The CLNW method consists in splitting the exponent as follows : a non-zero window will always be of length m, for a given parameter m, often equal to 4 in practice, and the zero windows are of variable length. For the exponentiation, precomputations have to be done for all the 2^{m-1} values of the m non-zero windows (with a bit 1 in low order since the parsing is done from the least significant bit to the most significant one).

The Variable Length version is an optimization of it and is more tricky to detail. The rule is to split the exponent into zero windows of length at least a given value and non-zero windows of length at most another given parameter. We can show that the number of leaking bits during a SPA attack will be the same for both techniques and so we only focus on the CLNW variant.

Figure J.2 details the sliding window exponentiation algorithm. Such a method assumes that the exponent is split into windows. This splitting may be done either with the m-ary, CLNW or VLNW method, depending on the splitting criteria used. The exponentiation just uses squarings and multiplications with precomputed values.

J.2.3 SPA Information leakage

To mount the attack described in section J.3, the underlying assumption will be that the attacker knows partial information on the exponent used during the RSA signature or decryption. To this end, we will assume that we can distinguish squares from multiplies.

– `Inputs` : x, e, N, m
– `Output` : $y = x^e \bmod N$

1. Compute and store x^w for all **odd** integer $w \in \{1, \ldots, 2^m - 1\}$
2. Parse e into zero and non-zero windows F_i of length $L(F_i)$ at most m for the non-zero windows and for $i = 0, 1, \ldots k-1$. The parsing algorithm may be CLNW or VLNW.
3. $y \leftarrow x^{F_{k-1}} \bmod N$ (which is a precomputed value)
4. for $i = k-2$ downto 0
   – $y \leftarrow y^{2^{L(F_i)}} \bmod N$
   – if $F_i \neq 0$, then $y \leftarrow y \cdot x^{F_i} \bmod N$
5. `return` y

FIGURE J.2: The Sliding Window Algorithm

The optimized exponentiation algorithms such as m-ary, CLNW or VLNW leak some information about the exponent. For example, the CLNW algorithm, as described in section J.2.2, consists in splitting the exponent into non-zero windows of fixed length. For all these windows, some precomputations are made to reduce the total cost of the exponentiation. If no protection against SPA attacks is used, an attacker may be able to distinguish the squaring operations from the multiplications. Each time the number of squarings is greater than the length of the window, the attacker can deduce that there are some 0 bits in the exponent. The position is deduced from the total number of previous multiplications and squarings. When a multiplication is detected, the attacker knows that there is a non-zero window of exactly m bits. In this window, the lowest order bit is 1 and the other ones are unknown. Thus, in the worst case, when there are are only non-zero windows, the attacker learns one bit of the exponent over m. These bits are the least significant ones (equal to 1) of the non-zero windows.

Thus, if $m = 4$, the attacker learns 25% of the bits of the exponent in the worst case. In practice, we obtain 40% of the bits of each randomized exponent. For $m = 3$, we obtain 50% of these bits.

For the attack to be successful in the case $e = 2^{16}+1$, the attacker has to obtain a given number of windows of two bits. Note that if the attacker learns 3 consecutive bits, the two overlapping windows of two bits can be used in the attack. Simulations show that for a 1024-bit modulus, the CLNW methods for parameter $m = 4$ may leak 200 such windows if squarings and multiplies are distinguishable. For 2048-bit modulus, 400 windows are obtained. For $m = 3$, we obtain 250 2-bit windows for 1024-bit modulus and 500 for 2048 ones.

J.3 Recovering a private RSA exponent from partial information on randomized versions of it

We focus in this section on the special case $e = 3$. In this context, additional and free information can be deduced from the public key. This gives the attacker the knowledge of some bits on the private key "for free". Although this does not allow an adversary to break RSA cryptosystems in general, such information is very useful when combined with a side channel attack on some particular implementation of the exponentiation.

J.3.1 Free information

Let N be an RSA modulus, e the public exponent and d the private one.

The first remark is that the modulus N is a good approximation of $\varphi(N) = (p-1) \times (q-1) = N - p - q + 1$ on essentially the $n/2$ most significant bits. Since the number of these bits depends on a carry propagation, with high probability, the $n/2 - 10$ bits of high order of $\varphi(N)$ are those of N. To simplify, we consider that the $n/2$ bits of high order of $\varphi(N)$ are known and equal to those of N.

Secondly, when $e = 3$, the $n/2$ most significant bits of d are also known. Indeed, d satisfies the relation

$$ed = 1 + k\varphi(N) \tag{J.1}$$

for some positive integer k. Let us choose a representative of d in $[0, \varphi(N) - 1]$. Since $k \times \varphi(N) = ed - 1 < ed$, then $k < e$: if $e = 3$, then $k = 1$ or $k = 2$. In fact, 3 divides neither p nor q and since 3 is invertible mod $\varphi(N)$, 3 also divides neither $p - 1$ nor $q - 1$. Thus, $p \neq 0 \bmod 3$, $p - 1 \neq 0 \bmod 3$ (resp. for q), and finally, $p = 2 \bmod 3$ and $q = 2 \bmod 3$. Consequently, $\varphi(N) = 1 \bmod 3$. Finally, since $k = -1/\varphi(N) \bmod 3$, then $k = 2 \bmod 3$, and finally $k = 2$. Therefore,

$$3d - 2\varphi(N) = 1 \tag{J.2}$$

and

$$\tilde{d} = \left\lfloor \frac{1 + kN}{e} \right\rfloor = \left\lfloor \frac{1 + 2N}{3} \right\rfloor$$

is a good approximation of d on the half bits of high order. Equation (J.2) will be extensively used in the cryptanalysis described above.

J.3.2 Recovering the RSA private key

We consider the countermeasure consisting in randomizing the private exponent d. Thus for each exponentiation, an equivalent exponent d_i is first computed as $d_i = d + \lambda_i \times \varphi(N)$, for a random value λ_i of ℓ bits. Typically, $\ell = 20$ or $\ell = 32$. Furthermore, an optimized exponentiation algorithm, such as the CLNW or the VLNW method, is supposed to be used. In this context, as described in section J.2.3, we suppose that the attacker knows a fraction $1/r$ of the bits of the private exponents used, randomly distributed amongst the $n + \ell$ bits of each exponent. We also suppose that these bits are available for ω different exponents d_i. The position of the known bits differ from one exponent to another. These bits are obtained by signing or decrypting ω messages whose value does not matter for the attack. This is a model for the side channel attack.

In the rest of this paper, the following notations will be used :

- for an integer x of n bits, the i-th bit of x is denoted by $x[i]$. The integer x can then be written as an n-bit string $x = x[n-1]x[n-2]\dots x[0]$;
- for a randomized exponent d_i of n bits, $[d_i]$ is a vector of length n such that for all $j \in \{0, \dots, n-1\}$:

$$\begin{aligned}[d_i][j] &= d_i[j] \text{ if the bit } d_i[j] \text{ is known} \\ &= 2 \text{ otherwise}\end{aligned}$$

- for a vector $[d_i]$ of length n and integers a and b such that $0 \leq a \leq b \leq n-1$, $[d_i]_{a,b}$ is the extracted vector for the positions a to b;
- for integers x and d_i of n bits, we write $[d_i] \doteq x$ if d_i and x matches on all the known bits of d_i. That is, for all $0 \leq j \leq n-1$ such that $[d_i][j] \neq 2$, we have $d_i[j] = x[j]$.

For example, for an 8-bit value $x = 01010101 = x[7]x[6]\dots x[1]x[0]$ for which the bits known are in positions 1, 4, 5 and 7, $[x] = [0, 2, 0, 1, 2, 2, 0, 2]$ and $[x]_{3,6} = [2, 0, 1, 2]$.

We first show how partial knowledge of the randomized exponents d_i allows the attacker to recover the λ_i values used to generate them from d. We then show in a second step how to recover the entire private exponent d from the partial leakage on the randomized exponent and from the known bits of d.

Step 1 : recovering the λ_i.

The strategy to recover the random values λ_i used to mask the private exponent d is to use the known approximation of d and $\varphi(N)$ to compute all the possible values

$$\tilde{d_j} = \tilde{d} + j \times N$$

for all $j \in [0, 2^\ell - 1]$. On the $n/2$ high order bits, $\tilde{d_j}$ is equal to $d_j = d + j \times \varphi(N)$. Indeed, $d_j = \tilde{d_j} + j(p + q - 1)$ and since $j(p + q - 1)$ is a number of at most $(n/2 + \ell)$ bits, the two $(n + \ell)$-bit values have near half of the most significant bits in common with high probability. Only if a carry propagates, some bits will not be equal. However, if two random bitstrings are added, a carry will be absorbed with probability $1/4$ at a step. Consequently, with probability $1 - (3/4)^{10} \approx 0.94$, the carry coming from the least significant half bits will not propagate after the $(n/2 + \ell + 10)$-th bit.

Given these 2^ℓ values and the known bits of the ω randomized exponents d_i, the attacker is now able to recover the corresponding λ_i values. For each value d_i, he knows a ratio $1/r$ of the bits. In particular this applies for the $n/2$ high order ones. He then looks for a matching value from the computed $\tilde{d_j}$ on these bits. When he finds j such that d_i equals $\tilde{d_j}$ on the known bits, he deduces that $\lambda_i = j$. The detailed algorithm is given in figure J.3.

Some optimizations may be implemented depending on the value ℓ, and the best time-memory trade-off for the attacker. However, as long as the random values λ are relatively small, for example of at most 20 bits, the exhaustive search of figure J.3 is clearly practical.

At the end of this step, the attacker has thus recovered each random value used to randomize the private exponent for ω RSA executions.

Let us now present some analysis of the success probability of the first step. For each given d_i, we compare it with the 2^ℓ values of $\tilde{d_j}$. Let us denote by Bad_i the event "a bad value λ is associated with d_i". If we assume that the bitstrings d_i and $\tilde{d_j}$ are uniformly distributed, we match a false j to λ_i with probability less than $(1/2)^{(n/2-\alpha)/r}$ where α depends on the carry propagation during the computation of $\tilde{d_j}$. As seen above, α may be upper bounded by 10 with high probability.

– `Inputs` : $[d_i]_{n/2+\ell,n+\ell}$ the known bits of high order of d_i for all $i \in \{1,\ldots,\omega\}$
– `Outputs` : the value λ_i such that $d_i = d + \lambda_i \times \varphi(N)$, for all $i \in \{1,\ldots,\omega\}$

1. For $j = 0$ to $2^\ell - 1$, $\tilde{d}_j \leftarrow \tilde{d} + j \times N$
2. For $i = 1$ to ω,
   $j \leftarrow 0$
   While $(j < 2^\ell)$
      If $[d_i]_{n/2+\ell+10,n+\ell} \doteq \tilde{d}_{j\,n/2+\ell+10,n+\ell}$ then $\lambda_i \leftarrow j$, break ;
      else $j \leftarrow j + 1$;
3. Return λ_i for all $i \in \{1,\ldots,\omega\}$.

FIGURE J.3: The attacker strategy to recover the λ_i corresponding to each d_i

As we have 2^ℓ comparisons corresponding to all the $\tilde{d}_j$, the probability of Bad_i is upper bounded by $2^\ell/2^{(n/2-\alpha)/r}$. Therefore, we get

$$\Pr[\exists i\ 1 \le i \le \omega : \mathsf{Bad}_i] \le \frac{2^\ell \omega}{2^{(n/2-\alpha)/r}}$$

For $n = 1024$, $r = 5$, $\ell = 32$, $\alpha = 10$ and $\omega \approx 64$, we get a probability of a good association for each d_i of $1-1/2^{63}$. Such an estimation does not take into account imperfect input data.

Step 2 : recovering $\varphi(N)$ and d.

The attacker's goal is now to recover the entire private key. To this end, he makes an exhaustive search, 8 bits per step, on the bits of $\varphi(N)$. He will now use the known *least significant bits* of d_i to recover d.

First, the attacker recovers the 8 least significant bits of $\varphi(N)$. This is performed by guessing $\varphi(N) \bmod 2^8$. From this guess, he computes the corresponding guess for $d \bmod 2^8$ from the equation J.2 :

$$d \bmod 2^8 = \frac{1 + 2\varphi(N)}{3} \bmod 2^8$$

Then with high probability there exists i s.t. some of the 8 least significant bits of d_i are known from the side channel attack. For this value d_i, the corresponding λ_i gives us some constraints on the low order bits of the value $d + \lambda_i \times \varphi(N) \bmod 2^8$. If the constraints on the corresponding d_i value cannot be met, another guess for $\varphi(N) \bmod 2^8$ is made. Otherwise, if for all $i \in \{1,\ldots,\omega\}$, no incompatibility has been discovered, the guess is the good one with high probability. The attack can then be extended with a guess for $\varphi(N) \bmod 2^{16}$ and so on. Figure J.4 details the algorithm to recover $\varphi(N) \bmod 2^{8k}$ from $\varphi(N) \bmod 2^{8(k-1)}$.

Note that in practice the attacker should deal with imperfect input data since these data are collected in a side channels context. Thus, candidates that match a sufficiently high fraction of these data should be accepted : this may be done by implementing a more complex version of the Boolean function OK.

We need to estimate the average number of false candidates at each step. We have 2^8 values for each $\bar{d}$ and we have $8/r$ bits on each 8-bit window for each $\bar{d}_i$ where $1/r$

- `Inputs` :
  - $\{([d_i], \lambda_i)\}_{1 \leq i \leq \omega}$ the list of the known bits for each d_i and the corresponding λ_i value
  - a candidate for $\varphi(N) \bmod 2^{8(k-1)}$
- `Output` : a list of candidates for $\varphi(N) \bmod 2^{8k}$

1. For $j = 0$ to $2^8 - 1$,
   (a) $y_j \leftarrow \varphi(N) \bmod 2^{8(k-1)} + j \cdot 2^{8(k-1)}$;
   `/* ` y_j `is a candidate value for` $\varphi(N) \bmod 2^{8k}$ `*/`
   (b) $\bar{d} \leftarrow \frac{1+2y_j}{3} \bmod 2^{8k}$;
   `/* ` $\bar{d}$ `is the corresponding candidate for` $d \bmod 2^{8k}$ `*/`
   (c) $OK \leftarrow$ `true` ;
   (d) $i \leftarrow 1$;
   (e) While ($OK =$ `true`) and ($i \leq \omega$)
   $\bar{d}_i \leftarrow \bar{d} + \lambda_i \times y_j \bmod 2^{8k}$;
   if $[d_j]_{0,8k-1} \doteq \bar{d}_i$ then $i \leftarrow i + 1$;
   else $OK \leftarrow$ `false` ;
   (f) if $OK =$ `true`, add y_j to the list of candidates for $\varphi(N) \bmod 2^{8k}$;
2. Return the list of candidates for $\varphi(N) \bmod 2^{8k}$

FIGURE J.4: The attacker strategy to recover $\varphi(N) \bmod 2^{8k}$ from $\varphi(N) \bmod 2^{8(k-1)}$

is the ratio of known bits deduced from the side channel attack. As the correct value for $\bar{d}$ allows ω correct values $\bar{d}_i$ for all i to be computed, then on average the number of false candidates is $\left(1/2^{8/r}\right)^{\omega} \times 2^8$ if all the experiments are independent and the bitstrings uniformly distributed. Thus, the average number of candidates tends to 1 as the number of false candidates tends to 0 and the correct candidate matches the input data, or eventually almost fit the input data.

J.4 Extension for $e = 2^{16} + 1$

In case $e = 3$, one knows that $k = 2$. For each measure using a random value λ, as shown in previous section, some bits in the upper half of

$$\left\lfloor \frac{1 + kN}{e} \right\rfloor + \lambda N \tag{J.3}$$

are known : this allows us to retrieve λ with an exhaustive search. For other values of e, this approach cannot work directly as k is not known anymore. In this section, we show how to extract k *and* λ from only **one** measure yielding some bits of the randomized exponent, when e is not too large, the typical case being $e = 2^{16} + 1$. Once k is found, the attack can proceed exactly as described in Step 2 of the attack of section J.3.

J.4.1 Finding k and λ by exhaustive search

The value (J.3) can be used to perform a direct exhaustive search of k and λ from one exponentiation measure. One has $0 < k < e$ and $0 \leq \lambda < 2^\ell$: if e is u-bit long, there are $2^{u+\ell}$ candidates to try, and the exhaustive search yields only the correct values with good probability if the number of known bits in equation (J.3) is above $u + \ell$. For the typical values of $u = 16$ and $\ell = 20$, this approach requires to perform 2^{36} additions of large integers, assuming the values of λN for $0 \leq \lambda < 2^\ell$ and $\frac{kN}{e}$ for $0 < k < e$ are precomputed. The aim is to recover k more efficiently.

J.4.2 Finding Matching Pairs of Values of k and λ

From now on, we consider a unique measure of an RSA signature or decryption with randomized exponent. Let δ denote the randomized exponent used during the exponentiation considered. One has :

$$\delta = d + \lambda \times \varphi(N)$$

As before, the most significant half $U(d)$ of the private exponent d is equal to $U\left(\left\lfloor \frac{1+kN}{e} \right\rfloor\right)$ except maybe on a few least significant bits. $U(\delta)$ can likewise be approximated by $U\left(\left\lfloor \frac{1+kN}{e} \right\rfloor\right) + U(\lambda N)$. Our goal is to recover k and λ.

If $U(\delta)$ were completely known, the exhaustive search on both k and λ could be transformed it into a list matching problem : indeed, correct values of k and λ correspond to matching elements in the lists

$$L_1 = \left\{ U(\delta) - U\left(\left\lfloor \frac{1+kN}{e} \right\rfloor\right) \;\middle|\; 0 < k < e \right\} \text{ and } L_2 = \left\{ U(\lambda N) \mid 0 \leq \lambda < 2^\ell \right\}$$

However, since only *some* bits of δ are known, some further work is required to find matching elements. In the next paragraphs, we show how to associate with each candidate value of k a partially known value for $\delta - \left\lfloor \frac{1+kN}{e} \right\rfloor$, and then how to find matches between the list of these partially known values and L_2.

Step 1a : Compute Partial Values for $\delta - \left\lfloor \frac{1+kN}{e} \right\rfloor$.

In the following, $b(k)$ denotes $\left\lfloor \frac{1+kN}{e} \right\rfloor$.

For each candidate value of k, some bits of $\delta - b(k)$ can be computed. Indeed, assume that two consecutive bits δ_i, δ_{i+1} of δ are known as a result a side-channel attack like the one of paragraph J.2.3. Let b_i and b_{i+1} the corresponding bits of $b(k)$, and c_i, c_{i+1} the corresponding carry bits in $\delta - b(k)$. The bits δ_i, δ_{i+1}, b_i, b_{i+1} are known while the carries are unknown. The subtraction looks as follows :

$$\begin{array}{r} \delta_{i+1} \quad \delta_i \\ -\, b_{i+1} \overset{c_{i+1}}{\leftarrow} b_i \overset{c_i}{\leftarrow} \\ \hline \cdots \quad \cdots \end{array}$$

Assume that $b_i = 1 \oplus \delta_i$. Then one has :

$$\begin{array}{r} \delta_{i+1} \quad 0 \\ -\, \overset{\cdots}{\leftarrow} b_{i+1} \overset{1}{\leftarrow} 1 \overset{c_i}{\leftarrow} \\ \hline 1 \oplus \delta_{i+1} \oplus b_{i+1} \quad 1 \oplus c_i \end{array} \quad \text{or} \quad \begin{array}{r} \delta_{i+1} \quad 1 \\ -\, \overset{\cdots}{\leftarrow} b_{i+1} \overset{0}{\leftarrow} 0 \overset{c_i}{\leftarrow} \\ \hline \delta_{i+1} \oplus b_{i+1} \quad 1 \oplus c_i \end{array}$$

Therefore **whenever $b_i = 1 \oplus \delta_i$, the $(i+1)$-th bit of $\delta - b$ is equal to $b_i \oplus \delta_{i+1} \oplus b_{i+1}$ which is a known value**.

To compute partial values for $\delta - b(k)$, first mark the (possibly overlapping) windows of two consecutive known bits in the upper half of δ. Assume there are v such windows.

For each value of k in $[1, e-1]$, compute the value $b(k) = \left\lfloor \frac{1+kN}{e} \right\rfloor$. Depending on the bits of $b(k)$ aligned with the marked windows in δ, some bits of $\delta - b(k)$ can be computed according to the rules above : in each 2-bit window, the most significant bit of $\delta - b(k)$ can be computed with probability 1/2, according to $b_i = 1 \oplus \delta_i$. A sequence (s_k) of v known or unknown bits in $\delta - b(k)$ is therefore obtained.

Step 1b : Find Matches for Partial Values.

Associate to each value of λ the sequence t_λ of the values of the most significant bits of λN in the targeted 2-bit windows of δ (remember that there are v such windows). The correct value for k and λ yields a match between t_λ and the known bits in s_k. If there are sufficiently many windows, only the correct values gives a match.

For each value of k, let $L(k)$ denote the set of all λ such that t_λ matches s_k. Assuming $L(k)$ can be built efficiently, the exhaustive search for (k, λ) proposed in subsection J.4.1 can be improved by adding an early elimination step where pairs (k, λ) s.t. $\lambda \notin L(k)$ are discarded. If $L(k)$ is small enough, this reduces the complexity of the exhaustive search. We therefore focus on efficiently computing the set $L(k)$.

A direct approach to the construction of $L(k)$ consists in considering every possible value of the unknown bits of s_k. For each of them, the set of the matching t_λ can be computed. Since each of the v bits in s_k is known with probability 1/2, the number of possible values for the unknown bits in s_k is the product of v independent random variables that are equal to 2 with probability 1/2 and to 1 with probability 1/2. This number is therefore equal on average to $(3/2)^v$.

For $u = 16$, $v = 40$, there are about $2^{16} \times (3/2)^{40} \approx 2^{39.4}$ completions of the s_k, for all values of k, $0 < k < e$. Therefore, whatever the method used to find the corresponding t_λ for each of these completions, at least $2^{39.4}$ operations are required to build all lists $L(k)$ this way.

This approach can however be refined by splitting s_k into subpieces before exploring the possible values of the unknown bits. From now on, we assume that $v = 2\ell$; the attack is even faster for higher values v which correspond to cases where more information is available.

First, precompute the lists $L_l(\alpha) = \{\lambda \mid \text{the left half of } t_\lambda \text{ is equal to } \alpha\}$ for $0 \leq \alpha \leq 2^\ell$ and the lists $L_r(\beta) = \{\lambda \mid \text{the right half of } t_\lambda \text{ is equal to } \beta\}$ for $0 \leq \beta \leq 2^\ell$. This requires $2 \times 2^\ell$ operations on large integers.

Then for a candidate k, if $\alpha_1, \ldots, \alpha_n$ (resp. $\beta_1, \ldots, \beta_m$) are the values of the left (resp. right) half of s_k obtained by filling the unknown bits with any possible value,

$$L(k) = \left[\bigcup_{i=1}^{n} L_l(\alpha_i)\right] \cap \left[\bigcup_{i=1}^{m} L_r(\beta_i)\right]$$

On average, $n \approx m \approx (3/2)^{v/2}$, and for any i, $\#L_l(\alpha_i) \approx \#L_r(\beta_i) \approx 2^{\ell - v/2} = 1$. Using suitable data structure (of size $2^{v/2}$) to be able to compute an intersection in constant time, the formula above can therefore be evaluated using $2 \times (3/2)^{v/2}$ constant-size operations. This means that all the lists $L(k)$ can be built using only $2^u \times 2 \times (3/2)^{v/2}$ operations.

Modulus size	ℓ, size of random	$1/r$, ratio of partially known information	attack success
512	20	1/16	no
1024	20	1/16	yes
	32	1/16	no
2048	20	1/32	yes
	32	1/64	yes

FIGURE J.5: Practical results for $e = 3$.

For $u = 16$, $\ell = 20$, $v = 2\ell = 40$, the total complexity is $2^{28.7}$ operations. One can show that cutting s_k in two halves is optimal when $v = 2\ell$.

Assuming that the t_λ and the completions of the s_k are random, the birthday paradox shows that the average number of elements in $L(k)$ is $\frac{(3/2)^v \times 2^\ell}{2^v}$. With $\ell = 20$, $v = 2\ell$, $\#L(k) \approx 2^{3.4}$. Therefore after the above early elimination step, $2^{19.4}$ pairs (k, λ) must be considered if $u = 16$, compared to 2^{36} pairs before the elimination step.

Overall, this improved attack retrieves k using $2^{28.7}$ constant-size operations and around 2^{20} operations on large integers. We implemented the attack ; it runs in about one minute on an average PC.

In practice, as shown in section J.2.3, the number of windows available is far above what is needed : with a 1024-bit exponent and the exponentiation algorithm of figure J.2 with windows of size 4, one has approximately $v = 100$ windows of two known consecutive bits in the upper half of δ. This extra information can be taken into account to eliminate more pairs (k, λ). With v large enough, this filters out all the wrong pairs, thereby eliminating the need for an exhaustive search phase. On the other side, the complexity of the pair elimination phase is linear in v.

J.5 Practical results

There are two limits for this attack : the first one is to have enough information on one curve to be able to recover only one λ for each curve, the second one is to have enough information on all the curves to recover only one possibility for the least significant bits. The following tables will give some examples where the attack is feasible or not. We can note that the attack is more efficient on larger modulus size. Table J.5 gives the results in the case $e = 3$. In that case, only a ratio of bits has to be known in each randomized exponent. In practice, if side channel attack is possible on a CLNW or VLNW splitting method, we may obtain a better ratio than detailed in the table.

Table J.6 gives the results for $e = 2^{16} + 1$. In that case, as explained in section J.4, the attack has better complexity if 2-bit windows are obtained for one randomized exponent. Such information may be obtained with the CLNW or VLNW splitting algorithms. For the optimized method to be more efficient than exhaustive search, the number of 2-bit windows should be twice the length of the random value λ used to randomized the private exponent.

The fifth column is computed by using the formula $n/(2r) \gg \ell$. For each parameter, 50 curves are sufficient in practice, without considering imperfect input data.

In conclusion we can see that for classical size and reasonable information leaking, the attack is feasible and of low complexity.

Modulus size	ℓ, size of random	number of 2-bit windows	$1/r$, ratio of partially known information	attack success
512	20	40	1/16	no
1024	20	40	1/16	yes
	32	64	1/16	yes
2048	32	64	1/32	yes

FIGURE J.6: Practical results for $e = 2^{16} + 1$.

J.6 Appendix : When few bits are missing

In this appendix, we show that when the number of missing bits is small, we can recover missing bits of d by using a discrete log based algorithm. Stinson describes and analyzes several algorithms due to Heiman and Odlyzko and Coppersmith in [275]. Let m be the number of missing bits of $x = \log_\alpha \beta$ and t is the Hamming weight of x. Heiman and Odlyzko describe a meet-in-the-middle attack. We search Y_1 and $Y_2 \subseteq \mathbb{Z}_m$ such that $\alpha^{\mathsf{val}(Y_1)} = \beta(\alpha^{\mathsf{val}(Y_2)})^{-1} \bmod N$ where $\mathsf{val}(Y_i) = \sum_{j \in Y_i} 2^j$ by ranging through all Y_1 and Y_2 such that $|Y_1| = |Y_2| = t/2$. As there are $\binom{m}{t/2}$ such sets Y_1 and Y_2, the space and time complexity of the attack is of order $O(\binom{m}{t/2})$. Moreover, if we have only an upper bound t' on t, we have to run through all $t = 1$ to $t = t'$ and the time complexity becomes $O(\sum_{t=1}^{t'} \binom{m}{t/2}) = O(t'\binom{m}{t'/2})$.

Coppersmith's algorithm, described in [275], allows one to lower the time complexity to $O(m\binom{m/2}{t/2})$ and the space complexity to $O(\binom{m/2}{t/2})$. The idea is to use an (m, t)-splitting system for $\mathbb{Z}_m$. Such combinatorial structure is a pair $(X, \mathcal{B})$ with the following properties :

1. $|X| = m$, and $\mathcal{B}$ is a set of subsets of size $m/2$ of X, called *blocks*
2. for every $Y \subseteq X$ s.t. $|Y| = t$, there exists a block $B \in \mathcal{B}$ s.t. $|B \cap Y| = t/2$

Coppersmith shows that there exists an (m, t)-splitting system of size $m/2$. Therefore by picking $Y_1 \subseteq B_i$ for all $B_i \in \mathcal{B}$ and $Y_2 \in \mathbb{Z}_m \setminus B_i$ for any t-set Y_2 in $\mathbb{Z}_m \setminus B_i$ the same algorithm finds the matching in time $O(\binom{m/2}{t/2})$.

The last algorithm can be adapted to work mod N where N is a RSA modulus and when the missing bits are not consecutive. The memory complexity is $O(m\binom{m/2}{t/2})$ where t is the number of 1 bits among m bits.

Consequently, if we assume that in the m missing bits, one of two are a one, then $t = m/2$. Therefore, the complexity is $O(m^2\binom{m/2}{m/4})$. Since, $\binom{N}{N/2} \approx \sqrt{2/\pi} \cdot 2^N$, then the complexity becomes $O(m^2 \cdot 2^{m/2})$, and in practice, we can only deal with $m \approx 128$.

Bibliographie

[1] Onur Aciiçmez, Billy Bob Brumley, and Philipp Grabher. New results on instruction cache attacks. In Stefan Mangard and François-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems, CHES 2010, 12th International Workshop*, volume 6225 of *Lecture Notes in Computer Science*, pages 110–124. Springer, 2010. 53

[2] Onur Aciiçmez, Çetin Kaya Koç, and Jean-Pierre Seifert. Predicting secret keys via branch prediction. In Masayuki Abe, editor, *Topics in Cryptology - CT-RSA 2007, The Cryptographers' Track at the RSA Conference 2007*, volume 4377 of *Lecture Notes in Computer Science*, pages 225–242. Springer, 2007. 53

[3] Adi Akavia. Finding significant fourier transform coefficients deterministically and locally. *Electronic Colloquium on Computational Complexity (ECCC)*, 15(102), 2008. 48

[4] J.-P. Allouche. Sur la complexité des suites infinies. *Bull. Belg. Math. Soc.*, 1:133–143, 1994. 138, 150

[5] Jean-Paul Allouche and Amir Sapir. Restricted towers of hanoi and morphisms. In *Developments in Language Theory*, pages 1–10, 2005.

[6] Hamid R. Amirazizi and Martin E. Hellman. Time-memory-processor trade-offs. *IEEE Transactions on Information Theory*, 34(3):505–512, 1988. 42, 156

[7] Elena Andreeva, Charles Bouillaguet, Pierre-Alain Fouque, Jonathan J. Hoch, John Kelsey, Adi Shamir, and Sébastien Zimmer. Second preimage attacks on dithered hash functions. In Nigel P. Smart, editor, *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, volume 4965 of *Lecture Notes in Computer Science*, pages 270–288. Springer, 2008. 42, 137

[8] Hendrik K. Lenstra Arjen K. Lenstra and Laszlo Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, 1982. 9, 186

[9] Sanjeev Arora and Rong Ge. Learning parities with structured noise. ECCC Archive, 2010.

[10] Krishna B. Athreya and Peter Ney. *Branching processes*. Springer-Verlag, Berlin, New York„ 1972. 110

[11] Daniel Augot, Matthieu Finiasz, and Nicolas Sendrier. A family of fast syndrome based cryptographic hash functions. In Ed Dawson and Serge Vaudenay, editors, *Progress in Cryptology - Mycrypt 2005, First International Conference on Cryptology in Malaysia*, volume 3715 of *Lecture Notes in Computer Science*, pages 64–83. Springer, 2005. 15

[12] M. Bardet, J.-C. Faugère, B. Salvy, and B.-Y. Yang. Asymptotic Behaviour of the Degree of Regularity of Semi-Regular Polynomial Systems. In *MEGA'05*, 2005. Eighth International Symposium on Effective Methods in Algebraic Geometry. 100

[13] Magali Bardet. *Étude des systèmes algébriques surdéterminés. Applications aux codes correcteurs et à la cryptographie.* PhD thesis, Université de Paris 6, Décembre 2006. 100

[14] Magali Bardet, Jean-Charles Faugère, and Bruno Salvy. On the complexity of Gröbner basis computation of semi-regular overdetermined algebraic equations. In *Proc. International Conference on Polynomial System Solving (ICPSS)*, pages 71–75, 2004.

[15] Magali Bardet, Jean-Charles Faugère, Bruno Salvy, and Bo-Yin Yang. Asymptotic behaviour of the degree of regularity of semi-regular polynomial systems. In *Proc. of MEGA 2005, Eighth International Symposium on Effective Methods in Algebraic Geometry*, 2005.

[16] Elad Barkan, Eli Biham, and Nathan Keller. Instant ciphertext-only cryptanalysis of gsm encrypted communication. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference*, volume 2729 of *Lecture Notes in Computer Science*, pages 600–616. Springer, 2003.

[17] Aurélie Bauer and Antoine Joux. Toward a rigorous variation of coppersmith's algorithm on three variables. In Moni Naor, editor, *Advances in Cryptology - EUROCRYPT 2007, 26th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, volume 4515 of *Lecture Notes in Computer Science*, pages 361–378. Springer, 2007.

[18] Mihir Bellare. New proofs for nmac and hmac: Security without collision-resistance. In Cynthia Dwork, editor, *Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference*, volume 4117 of *Lecture Notes in Computer Science*, pages 602–619. Springer, 2006. 154, 166

[19] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying hash functions for message authentication. In Neal Koblitz, editor, *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference*, volume 1109 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 1996. 153, 154

[20] Mihir Bellare, Anand Desai, E. Jokipii, and Phillip Rogaway. A concrete security treatment of symmetric encryption. In *FOCS*, pages 394–403, 1997. 37, 38

[21] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations among notions of security for public-key encryption schemes. In Hugo Krawczyk, editor, *Advances in Cryptology - CRYPTO '98, 18th Annual International Cryptology Conference*, volume 1462 of *Lecture Notes in Computer Science*, pages 26–45. Springer, 1998. 38

[22] Mihir Bellare, Joe Kilian, and Phillip Rogaway. The security of cipher block chaining. In Yvo Desmedt, editor, *Advances in Cryptology - CRYPTO '94, 14th Annual International Cryptology Conference*, volume 839 of *Lecture Notes in Computer Science*, pages 341–358. Springer, 1994.

[23] Mihir Bellare and Daniele Micciancio. A new paradigm for collision-free hashing: Incrementality at reduced cost. In *EUROCRYPT*, pages 163–192, 1997. 13

[24] Mihir Bellare and Phillip Rogaway. Optimal asymmetric encryption. In *EUROCRYPT*, pages 92–111, 1994. 174

[25] E. Berlekamp. Factoring polynomials over finite fields. *J. Bell System Tech.*, 46:1853–1859, 1967.

[26] E. R. Berlekamp, R. J. McEliece, and V. Tilborg. On the inherent intractability of certain coding problem. *IEEE Transactions on Information Theory*, 24:384–386, 1978. 128

[27] Dennis S. Bernstein. *Matrix mathematics. Theory, facts, and formulas. 2nd expanded ed.* Princeton, NJ: Princeton University Press. xxxix, 1139 p., 2009. 105, 118

[28] Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. On the indifferentiability of the sponge construction. In Nigel P. Smart, editor, *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, volume 4965 of *Lecture Notes in Computer Science*, pages 181–197. Springer, 2008. 49

[29] Eli Biham. Cryptanalysis of multiple modes of operation. *J. Cryptology*, 11(1):45–58, 1998. 36

[30] Eli Biham. Cryptanalysis of patarin's 2-round public key system with s boxes (2r). In *EUROCRYPT*, pages 408–416, 2000. 72

[31] Eli Biham, Alex Biryukov, and Adi Shamir. Cryptanalysis of skipjack reduced to 31 rounds using impossible differentials. In *EUROCRYPT*, pages 12–23, 1999. 48

[32] Eli Biham and Yaniv Carmeli. Efficient reconstruction of rc4 keys from internal states. In Kaisa Nyberg, editor, *Fast Software Encryption, 15th International Workshop, FSE 2008*, volume 5086 of *Lecture Notes in Computer Science*, pages 270–288. Springer, 2008. 54

[33] Eli Biham and Rafi Chen. Near-collisions of sha-0. In Matthew K. Franklin, editor, *Advances in Cryptology - CRYPTO 2004, 24th Annual International Cryptology Conference*, volume 3152 of *Lecture Notes in Computer Science*, pages 290–305. Springer, 2004. 49

[34] Eli Biham, Rafi Chen, Antoine Joux, Patrick Carribault, Christophe Lemuet, and William Jalby. Collisions of sha-0 and reduced sha-1. In Ronald Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, volume 3494 of *Lecture Notes in Computer Science*, pages 36–57. Springer, 2005. 49, 137

[35] Eli Biham and Adi Shamir. Differential cryptanalysis of des-like cryptosystems. In Alfred Menezes and Scott A. Vanstone, editors, *Advances in Cryptology - CRYPTO '90, 10th Annual International Cryptology Conference*, volume 537 of *Lecture Notes in Computer Science*, pages 2–21. Springer, 1991. 6, 32, 48

[36] Olivier Billet and Henri Gilbert. A traceable block cipher. In Chi-Sung Laih, editor, *Advances in Cryptology - ASIACRYPT 2003, 9th International Conference on the Theory and Application of Cryptology and Information Security*, volume 2894 of *Lecture Notes in Computer Science*, pages 331–346. Springer, 2003. 34, 91, 93

[37] Alex Biryukov, Christophe De Cannière, An Braeken, and Bart Preneel. A toolbox for cryptanalysis: Linear and affine equivalence algorithms. In Eli Biham, editor, *Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques*, volume 2656 of *Lecture Notes in Computer Science*, pages 33–50. Springer, 2003. 92

[38] Alex Biryukov and Adi Shamir. Structural cryptanalysis of sasas. In Birgit Pfitzmann, editor, *Advances in Cryptology - EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques*, volume 2045 of *Lecture Notes in Computer Science*, pages 394–405. Springer, 2001.

[39] Simon R. Blackburn, Douglas R. Stinson, and Jalaj Upadhyay. On the complexity of the herding attack and some related attacks on hash functions. Cryptology ePrint Archive, Report 2010/030, 2010. `http://eprint.iacr.org/`. 42

[40] Avrim Blum, Merrick L. Furst, Michael J. Kearns, and Richard J. Lipton. Cryptographic primitives based on hard learning problems. In Douglas R. Stinson, editor, *Advances in Cryptology - CRYPTO '93, 13th Annual International Cryptology Conference*, volume 773 of *Lecture Notes in Computer Science*, pages 278–291. Springer, 1994. 128

[41] Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *J. ACM*, 50(4):506–519, 2003. 45, 128, 130, 131

[42] Andrej Bogdanov. Pseudorandom generators for low degree polynomials. In Harold N. Gabow and Ronald Fagin, editors, *Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, pages 21–30. ACM, 2005. 101

[43] Andrej Bogdanov and Emanuele Viola. Pseudorandom bits for polynomials. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007)*, pages 41–51. IEEE Computer Society, 2007. 101

[44] Bela Bollobas. *Random Graphs*. Cambridge University Press, 2001. 78

[45] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the importance of checking cryptographic protocols for faults (extended abstract). In *EUROCRYPT*, pages 37–51, 1997. 60, 184

[46] Dan Boneh and Glenn Durfee. Cryptanalysis of rsa with private key. In *EUROCRYPT*, pages 1–11, 1999.

[47] Dan Boneh, Glenn Durfee, and Yair Frankel. An attack on rsa given a small fraction of the private key bits. In Kazuo Ohta and Dingyi Pei, editors, *Advances in Cryptology - ASIACRYPT '98, International Conference on the Theory and Applications of Cryptology and Information Security*, volume 1514 of *Lecture Notes in Computer Science*, pages 25–34. Springer, 1998. 17, 198

[48] Dan Boneh, Glenn Durfee, and Nick Howgrave-Graham. Factoring $n = p^r q$ for large r. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference*, volume 1666 of *Lecture Notes in Computer Science*, pages 326–337. Springer, 1999. 17

[49] Dan Boneh, Antoine Joux, and Phong Q. Nguyen. Why textbook elgamal and rsa encryption are insecure. In Tatsuaki Okamoto, editor, *Advances in Cryptology - ASIACRYPT 2000, 6th International Conference on the Theory and Application of Cryptology and Information Security*, volume 1976 of *Lecture Notes in Computer Science*, pages 30–43. Springer, 2000.

[50] J. Bonneau and I. Mironov. Cache-Collision Timing Attacks against AES. In *Cryptographic Hardware and Embedded Systems - CHES 2006: 8th International Workshop*, volume 4249 of *Lecture Notes in Computer Science*, pages 201–215. Springer, 2006. 53

[51] Jurjen N. Bos and Matthijs J. Coster. Addition chain heuristics. In Gilles Brassard, editor, *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference*, volume 435 of *Lecture Notes in Computer Science*, pages 400–407. Springer, 1990. 200

[52] Wieb Bosma, John J. Cannon, and Catherine Playoust. The Magma Algebra System I: The User Language. *J. Symb. Comput.*, 24(3/4):235–265, 1997. 114

[53] Charles Bouillaguet, Orr Dunkelman, Gaëtan Leurent, and Pierre-Alain Fouque. Another look at complementation properties. In Seokhie Hong and Tetsu Iwata, editors, *Fast Software Encryption, 17th International Workshop, FSE 2010*, volume 6147 of *Lecture Notes in Computer Science*, pages 347–364. Springer, 2010. 32, 33

[54] Charles Bouillaguet, Jean-Charles Faugère, Pierre-Alain Fouque, and Ludovic Perret. Differential-algebraic algorithms for the isomorphism of polynomials problem. Cryptology ePrint Archive, Report 2009/583, 2009. `http://eprint.iacr.org/`. 30

[55] Charles Bouillaguet, Pierre-Alain Fouque, Antoine Joux, and Joana Treger. A family of weak keys in hfe (and the corresponding practical key recovery). *J. of Mathemtical Cryptology*, 2010.

[56] Charles Bouillaguet, Pierre-Alain Fouque, and Sébastien Zimmer. Practical hash functions constructions resistant to generic second preimage attacks beyond the birthday bound. IPL soumission, 2010.

[57] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop*, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29. Springer, 2004. 55

[58] Bruno Buchberger. *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal.* PhD thesis, University of Innsbruck, 1965. 105

[59] Paul Camion and Jacques Patarin. The knapsack hash function proposed at crypto'89 can be broken. In *EUROCRYPT*, pages 39–53, 1991. 14

[60] Christophe De Cannière and Christian Rechberger. Finding sha-1 characteristics: General results and applications. In Xuejia Lai and Kefei Chen, editors, *Advances in Cryptology - ASIACRYPT 2006, 12th International Conference on the Theory and Application of Cryptology and Information Security*, volume 4284 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2006. 49

[61] Brice Canvel, Alain P. Hiltgen, Serge Vaudenay, and Martin Vuagnoux. Password interception in a ssl/tls channel. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference*, volume 2729 of *Lecture Notes in Computer Science*, pages 583–599. Springer, 2003.

[62] J. W. S. Cassels. *An Introduction to the Geometry of Numbers.* Springer, 1971. Second Edition. 9, 20

[63] Julien Cathalo, François Koeune, and Jean-Jacques Quisquater. A new type of timing attack: Application to gps. In Colin D. Walter, Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2003, 5th International Workshop*, volume 2779 of *Lecture Notes in Computer Science*, pages 291–303. Springer, 2003. 52

[64] Bletchley Park National Codes Centre. Colossus and the breaking of the lorenz cipher. `http://www.bletchleypark.org.uk/content/lorenzcipher.pdf`, 2010. 3

[65] Florent Chabaud and Antoine Joux. Differential collisions in sha-0. In Hugo Krawczyk, editor, *Advances in Cryptology - CRYPTO '98, 18th Annual International Cryptology Conference*, volume 1462 of *Lecture Notes in Computer Science*, pages 56–71. Springer, 1998. 49, 50

[66] Thomas Chardin, Pierre-Alain Fouque, and Delphine Masgana. Theoretical study of rc4 against cache timing attack. En soumission, 2010. 41, 53

[67] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412. Springer, 1999. 55

[68] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template attacks. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop*, volume 2523 of *Lecture Notes in Computer Science*, pages 13–28. Springer, 2003. 55, 58

[69] David Chaum. Blind signatures for untraceable payments. In *CRYPTO*, pages 199–203, 1982. 176

[70] Philippe Chose, Antoine Joux, and Michel Mitton. Fast correlation attacks: An algorithmic point of view. In Lars R. Knudsen, editor, *Advances in Cryptology - EUROCRYPT 2002, International Conference on the Theory and Applications of Cryptographic Techniques*, volume 2332 of *Lecture Notes in Computer Science*, pages 209–221. Springer, 2002. 14, 46

[71] Alan Cobham. Uniform tag seqences. *Mathematical Systems Theory*, 6(3):164–192, 1972. 150

[72] Alan Cobham. Uniform tag sequences. *Mathematical Systems Theory*, 6(3):164–192, 1972. 42

[73] Henri Cohen. *A Course in Computational Algebraic Number Theory.* Springer, 2000. 74

[74] Scott Contini and Yiqun Lisa Yin. Forgery and partial key-recovery attacks on hmac and nmac using hash collisions. In Xuejia Lai and Kefei Chen, editors, *Advances in Cryptology - ASIACRYPT 2006, 12th International Conference on the Theory and Application of Cryptology and Information Security*, volume 4284 of *Lecture Notes in Computer Science*, pages 37–53. Springer, 2006. 49, 154, 155, 156, 157, 158, 159, 160, 161, 164

[75] Don Coppersmith. Small solutions to polynomial equations, and low exponent rsa vulnerabilities. *J. Cryptology*, 10(4):233–260, 1997. 16, 17

[76] Jean-Sébastien Coron. Resistance against differential power analysis for elliptic curve cryptosystems. In Çetin Kaya Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems, First International Workshop, CHES'99*, volume 1717 of *Lecture Notes in Computer Science*, pages 292–302. Springer, 1999. 55, 58, 174, 176, 177

[77] Jean-Sebastien Coron and Antoine Joux. Cryptanalysis of a Provably Secure Cryptographic Hash Function. Cryptology ePrint Archive, Report 2004/013, 2004. `http://eprint.iacr.org/`. 15

[78] Jean-Sébastien Coron and Alexander May. Deterministic polynomial-time equivalence of computing the rsa secret key and factoring. *J. Cryptology*, 20(1):39–50, 2007.

[79] Matthijs J. Coster, Antoine Joux, Brian A. LaMacchia, Andrew M. Odlyzko, Claus-Peter Schnorr, and Jacques Stern. Improved low-density subset sum algorithms. *Computational Complexity*, 2:111–128, 1992. 11

[80] Nicolas Courtois. The security of hidden field equations (hfe). In David Naccache, editor, *Topics in Cryptology - CT-RSA 2001, The Cryptographer's Track at RSA Conference 2001*, volume 2020 of *Lecture Notes in Computer Science*, pages 266–281. Springer, 2001. 71

[81] Nicolas Courtois, Magnus Daum, and Patrick Felke. On the security of hfe, hfev- and quartz. In Yvo Desmedt, editor, *Public Key Cryptography - PKC 2003, 6th International Workshop on Theory and Practice in Public Key Cryptography*, volume 2567 of *Lecture Notes in Computer Science*, pages 337–350. Springer, 2003. 71

[82] Nicolas Courtois, Alexander Klimov, Jacques Patarin, and Adi Shamir. Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In Bart Preneel, editor, *Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques*, volume 1807 of *Lecture Notes in Computer Science*, pages 392–407. Springer, 2000. 73

[83] Nicolas Courtois and Willi Meier. Algebraic attacks on stream ciphers with linear feedback. In Eli Biham, editor, *Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques*, volume 2656 of *Lecture Notes in Computer Science*, pages 345–359. Springer, 2003.

[84] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Springer, 2002. 6, 31, 33

[85] Ivan Damgård. A design principle for hash functions. In Gilles Brassard, editor, *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference*, volume 435 of *Lecture Notes in Computer Science*, pages 416–427. Springer, 1990. 14, 38, 139

[86] M. Daum. *Cryptanalysis of Hash Functions of the MD4-Family*. PhD thesis, Ruhr-University of Bochum, 2005. 156, 157

[87] R. D. Dean. *Formal Aspects of Mobile Code Security*. PhD thesis, Princeton University, January 1999. 39, 137, 138

[88] Bert den Boer and Antoon Bosselaers. Collisions for the compression function of md5. In *EUROCRYPT*, pages 293–304, 1993. 49, 154, 157, 161, 164

[89] Bert den Boer, Kerstin Lemke, and Guntram Wicke. A dpa attack against the modular reduction within a crt implementation of rsa. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop*, volume 2523 of *Lecture Notes in Computer Science*, pages 228–243. Springer, 2002. 184

[90] Jintai Ding. A new variant of the matsumoto-imai cryptosystem through perturbation. In Feng Bao, Robert H. Deng, and Jianying Zhou, editors, *Public Key Cryptography - PKC 2004, 7th International Workshop on Theory and Practice in Public Key Cryptography*, volume 2947 of *Lecture Notes in Computer Science*, pages 305–318. Springer, 2004. 22, 24, 28, 72, 73

[91] Jintai Ding, Vivien Dubois, Bo-Yin Yang, Chia-Hsin Owen Chen, and Chen-Mou Cheng. Could sflash be repaired? In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Part II - Track B: Logic, Semantics, and Theory of Programming & Track C: Security and Cryptography Foundations*, volume 5126 of *Lecture Notes in Computer Science*, pages 691–701. Springer, 2008. 22, 63

[92] Itai Dinur and Adi Shamir. Cube attacks on tweakable black box polynomials. In Antoine Joux, editor, *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, volume 5479 of *Lecture Notes in Computer Science*, pages 278–299. Springer, 2009.

[93] Hans Dobbertin. Cryptanalysis of md4. In Dieter Gollmann, editor, *Fast Software Encryption, Third International Workshop*, volume 1039 of *Lecture Notes in Computer Science*, pages 53–69. Springer, 1996. 49

[94] Hans Dobbertin. Cryptanalysis of md5 compress. In *In Rump Session of EuroCrypt ?96*, page 68442. Annoucement, 1996. 49

[95] Vivien Dubois. *Cryptanalyse de Schémas Multivariés*. PhD thesis, École normale supérieure, Paris, France, 2007. 102, 115

[96] Vivien Dubois, Pierre-Alain Fouque, Adi Shamir, and Jacques Stern. Practical cryptanalysis of sflash. In Alfred Menezes, editor, *Advances in Cryptology - CRYPTO 2007, 27th Annual International Cryptology Conference*, volume 4622 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2007. 26, 27, 34, 63, 81, 92, 94, 96, 97, 101

[97] Vivien Dubois, Pierre-Alain Fouque, and Jacques Stern. Cryptanalysis of sflash with slightly modified parameters. In Moni Naor, editor, *Advances in Cryptology - EUROCRYPT 2007, 26th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, volume 4515 of *Lecture Notes in Computer Science*, pages 264–275. Springer, 2007. iii, 25, 26, 27, 34, 63, 82, 83, 85, 87, 89, 92, 94, 96, 101

[98] Vivien Dubois, Louis Granboulan, and Jacques Stern. An efficient provable distinguisher for hfe. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Part II*, volume 4052 of *Lecture Notes in Computer Science*, pages 156–167. Springer, 2006. 31, 94, 101, 102

[99] Vivien Dubois, Louis Granboulan, and Jacques Stern. Cryptanalysis of hfe with internal perturbation. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *Public Key Cryptography - PKC 2007, 10th International Conference on Practice and Theory in Public-Key Cryptography*, volume 4450 of *Lecture Notes in Computer Science*, pages 249–265. Springer, 2007. 94

[100] Orr Dunkelman, Nathan Keller, and Adi Shamir. Improved single-key attacks on 8-round aes. Cryptology ePrint Archive, Report 2010/322, 2010. `http://eprint.iacr.org/`. 64

[101] Andrzej Ehrenfeucht, K. P. Lee, and Grzegorz Rozenberg. Subword Complexities of Various Classes of Deterministic Developmental Languages without Interactions. *Theor. Comput. Sci.*, 1(1):59–75, 1975. 150

[102] Jean-Charles Faugère and Antoine Joux. Algebraic cryptanalysis of hidden field equation (hfe) cryptosystems using gröbner bases. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference*, volume 2729 of *Lecture Notes in Computer Science*, pages 44–60. Springer, 2003. 24, 71, 90

[103] Jean-Charles Faugère, Ayoub Otmani, Ludovic Perret, and Jean-Pierre Tillich. Algebraic cryptanalysis of mceliece variants with compact keys. In Henri Gilbert, editor, *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, volume 6110 of *Lecture Notes in Computer Science*, pages 279–298. Springer, 2010. 63

[104] Jean-Charles Faugère and Ludovic Perret. Polynomial equivalence problems: Algorithmic and theoretical aspects. In Serge Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, volume 4004 of *Lecture Notes in Computer Science*, pages 30–47. Springer, 2006. 23, 92, 93, 98, 100, 105, 114

[105] Jean-Charles Faugère. A new efficient algorithm for computing gröbner bases (f4). *J. of Pure and Applied Algebra*, 139(1–3):61–88, 1999. 23, 30, 105

[106] W. Feller. *An Introduction to Probability Theory and Its Applications*, volume 1, chapter 12. John Wiley & Sons, 1971. 146

[107] Niels Ferguson, John Kelsey, Stefan Lucks, Bruce Schneier, Michael Stay, David Wagner, and Doug Whiting. Improved cryptanalysis of rijndael. In Bruce Schneier, editor, *Fast Software Encryption, 7th International Workshop, FSE 2000*, volume 1978 of *Lecture Notes in Computer Science*, pages 213–230. Springer, 2001.

[108] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1987. 11

[109] Matthieu Finiasz, Philippe Gaborit, and Nicolas Sendrier. Improved Fast Syndrome Based Cryptographic Hash Functions. In V. Rijmen, editor, *ECRYPT Hash Workshop 2007*, 2007. 15

[110] Pierre-Alain Fouque, Louis Granboulan, and Jacques Stern. Differential cryptanalysis for multivariate schemes. In Ronald Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, volume 3494 of *Lecture Notes in Computer Science*, pages 341–353. Springer, 2005. 24, 28, 71, 82, 94

[111] Pierre-Alain Fouque, Nick Howgrave-Graham, Gwenaëlle Martinet, and Guillaume Poupard. The insecurity of esign in practical implementations. In Chi-Sung Laih, editor, *Advances in Cryptology - ASIACRYPT 2003, 9th International Conference on the Theory and Application of Cryptology and Information Security*, volume 2894 of *Lecture Notes in Computer Science*, pages 492–506. Springer, 2003. 22

[112] Pierre-Alain Fouque, Antoine Joux, Gwenaëlle Martinet, and Frédéric Valette. Authenticated on-line encryption. In Mitsuru Matsui and Robert J. Zuccherato, editors, *Selected Areas in Cryptography, 10th Annual International Workshop, SAC 2003*, volume 3006 of *Lecture Notes in Computer Science*, pages 145–159. Springer, 2004. 37

[113] Pierre-Alain Fouque, Antoine Joux, and Guillaume Poupard. Blockwise adversarial model for on-line ciphers and symmetric encryption schemes. In Helena Handschuh and M. Anwar Hasan, editors, *Selected Areas in Cryptography, 11th International Workshop, SAC 2004*, volume 3357 of *Lecture Notes in Computer Science*, pages 212–226. Springer, 2004. 38

[114] Pierre-Alain Fouque, Sébastien Kunz-Jacques, Gwenaëlle Martinet, Frédéric Muller, and Frédéric Valette. Power attack on small rsa public exponent. In Louis Goubin and Mitsuru Matsui, editors, *Cryptographic Hardware and Embedded Systems - CHES 2006, 8th International Workshop*, volume 4249 of *Lecture Notes in Computer Science*, pages 339–353. Springer, 2006. 19, 56, 197

[115] Pierre-Alain Fouque, Reynald Lercier, Denis Réal, and Frédéric Valette. Fault attack onelliptic curve montgomery ladder implementation. In Luca Breveglieri, Shay Gueron, Israel Koren, David Naccache, and Jean-Pierre Seifert, editors, *Fifth International Workshop on Fault Diagnosis and Tolerance in Cryptography, 2008, FDTC 2008*, pages 92–98. IEEE Computer Society, 2008. 60

[116] Pierre-Alain Fouque and Gaëtan Leurent. Cryptanalysis of a hash function based on quasi-cyclic codes. In Tal Malkin, editor, *Topics in Cryptology - CT-RSA 2008, The Cryptographers' Track at the RSA Conference 2008*, volume 4964 of *Lecture Notes in Computer Science*, pages 19–35. Springer, 2008. 15

[117] Pierre-Alain Fouque, Gaëtan Leurent, and Phong Q. Nguyen. Full key-recovery attacks on hmac/nmac-md4 and nmac-md5. In Alfred Menezes, editor, *Advances in Cryptology - CRYPTO 2007, 27th Annual International Cryptology Conference*, volume 4622 of *Lecture Notes in Computer Science*, pages 13–30. Springer, 2007. 49, 50, 153

[118] Pierre-Alain Fouque, Gaëtan Leurent, Denis Réal, and Frédéric Valette. Practical electromagnetic template attack on hmac. In Christophe Clavier and Kris Gaj, editors, *Cryptographic Hardware and Embedded Systems - CHES 2009, 11th International Workshop*, volume 5747 of *Lecture Notes in Computer Science*, pages 66–80. Springer, 2009. 58

[119] Pierre-Alain Fouque, Gaëtan Leurent, and Phong Nguyen. Automatic Search of Differential Path in MD4. ECRYPT Hash Worshop – Cryptology ePrint Archive, Report 2007/206, 2007. `http://eprint.iacr.org/`. 156

[120] Pierre-Alain Fouque, Gilles Macario-Rat, and Jacques Stern. Key recovery on hidden monomial multivariate schemes. In Nigel P. Smart, editor, *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, volume 4965 of *Lecture Notes in Computer Science*, pages 19–30. Springer, 2008. 25, 26, 34, 91, 101

[121] Pierre-Alain Fouque, Gwenaëlle Martinet, and Guillaume Poupard. Attacking unbalanced rsa-crt using spa. In Colin D. Walter, Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2003, 5th International Workshop*, volume 2779 of *Lecture Notes in Computer Science*, pages 254–268. Springer, 2003. 21, 22, 56, 183

[122] Pierre-Alain Fouque, Gwenaëlle Martinet, and Guillaume Poupard. Practical symmetric on-line encryption. In Thomas Johansson, editor, *Fast Software Encryption, 10th International Workshop, FSE 2003*, volume 2887 of *Lecture Notes in Computer Science*, pages 362–375. Springer, 2003. 37

[123] Pierre-Alain Fouque, Gwenaëlle Martinet, Frédéric Valette, and Sébastien Zimmer. On the security of the ccm encryption mode and of a slight variant. In Steven M. Bellovin, Rosario Gennaro, Angelos D. Keromytis, and Moti Yung, editors, *Applied Cryptography and Network Security, 6th International Conference, ACNS 2008*, volume 5037 of *Lecture Notes in Computer Science*, pages 411–428, 2008. 43

[124] Pierre-Alain Fouque, Frédéric Muller, Guillaume Poupard, and Frédéric Valette. Defeating countermeasures based on randomized bsd representations. In Marc Joye and Jean-Jacques Quisquater,

editors, *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop*, volume 3156 of *Lecture Notes in Computer Science*, pages 312–327. Springer, 2004.

[125] Pierre-Alain Fouque and Guillaume Poupard. On the security of rdsa. In Eli Biham, editor, *Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques*, volume 2656 of *Lecture Notes in Computer Science*, pages 462–476. Springer, 2003. 11

[126] Pierre-Alain Fouque, Denis Réal, Frédéric Valette, and M'hamed Drissi. The carry leakage on the randomized exponent countermeasure. In Elisabeth Oswald and Pankaj Rohatgi, editors, *Cryptographic Hardware and Embedded Systems - CHES 2008, 10th International Workshop*, volume 5154 of *Lecture Notes in Computer Science*, pages 198–213. Springer, 2008. 57

[127] Pierre-Alain Fouque, Jacques Stern, and Sébastien Zimmer. Cryptanalysis of tweaked versions of smash and reparation. In Roberto Maria Avanzi, Liam Keliher, and Francesco Sica, editors, *Selected Areas in Cryptography, 15th International Workshop, SAC 2008*, volume 5381 of *Lecture Notes in Computer Science*, pages 136–150. Springer, 2009. 14

[128] Pierre-Alain Fouque and Frédéric Valette. The doubling attack - *hy upwards is better than downwards*. In Colin D. Walter, Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2003, 5th International Workshop*, volume 2779 of *Lecture Notes in Computer Science*, pages 269–280. Springer, 2003. 58, 173

[129] Atsushi Fujioka, Tatsuaki Okamoto, and Shoji Miyaguchi. Esign: An efficient digital signature implementation for smard cards. In *EUROCRYPT*, pages 446–457, 1991. 22

[130] Jason Fulman. Random matrix theory over finite fields. *Bull. Amer. Math. Soc. (N.S*, 39:51–85. 106, 119

[131] Giordano Fusco and Eric Bach. Phase transition of multivariate polynomial systems. *Mathematical. Structures in Comp. Sci.*, 19(1):9–23, 2009. 107

[132] Nicolas Gama and Phong Q. Nguyen. Predicting lattice reduction. In Nigel P. Smart, editor, *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, volume 4965 of *Lecture Notes in Computer Science*, pages 31–51. Springer, 2008. 9

[133] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *CRYPTO*, pages 10–18, 1984. 81, 174

[134] M. R. Garey and D. S. Johnson. *Computers and Intractability : A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979. Series of Books in the Mathematical Sciences. 22, 81, 105

[135] Jochen Geiger. Elementary new proofs of classical limit theorems for Galton-Watson processes. *J. Appl. Probab.*, 36(2):301–309, 1999. 101, 111, 119, 120

[136] Irina Gheorghiciuc. *The Subword Complexity of Finite and Infinite Binary Words*. PhD thesis, University of Pennsylvania, September 2004. 145

[137] Henri Gilbert and Marine Minier. Cryptanalysis of sflash. In Lars R. Knudsen, editor, *Advances in Cryptology - EUROCRYPT 2002, International Conference on the Theory and Applications of Cryptographic Techniques*, volume 2332 of *Lecture Notes in Computer Science*, pages 288–298. Springer, 2002. 72, 82, 92

[138] Henri Gilbert, Matt Robshaw, and Herve Sibert. An active attack against hb+ - a provably secure lightweight authentication protocol. Cryptology ePrint Archive, Report 2005/237, 2005. `http://eprint.iacr.org/`. 130

[139] Marc Girault, Guillaume Poupard, and Jacques Stern. On the fly authentication and signature schemes based on groups of unknown order. *J. Cryptology*, 19(4):463–487, 2006.

[140] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing*, pages 25–32. ACM, 1989.

[141] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design (extended abstract). In *27th Annual Symposium on Foundations of Computer Science*, pages 174–187. IEEE, 1986. 100

[142] Louis Goubin. A refined power-analysis attack on elliptic curve cryptosystems. In Yvo Desmedt, editor, *Public Key Cryptography - PKC 2003, 6th International Workshop on Theory and Practice in Public Key Cryptography*, volume 2567 of *Lecture Notes in Computer Science*, pages 199–210. Springer, 2003. 55, 177

[143] Louis Goubin and Jacques Patarin. Des and differential power analysis (the "duplication" method). In Çetin Kaya Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems, First International Workshop*, volume 1717 of *Lecture Notes in Computer Science*, pages 158–172. Springer, 1999. 55

[144] Louis Granboulan, Antoine Joux, and Jacques Stern. Inverting hfe is quasipolynomial. In Cynthia Dwork, editor, *Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference*, volume 4117 of *Lecture Notes in Computer Science*, pages 345–356. Springer, 2006. 24

[145] JaeCheol Ha and Sang-Jae Moon. Randomized signed-scalar multiplication of ecc to resist power attacks. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop*, volume 2523 of *Lecture Notes in Computer Science*, pages 551–563. Springer, 2003. 177

[146] Helena Handschuh and Howard M. Heys. A timing attack on rc5. In Stafford E. Tavares and Henk Meijer, editors, *Selected Areas in Cryptography '98, SAC'98*, volume 1556 of *Lecture Notes in Computer Science*, pages 306–318. Springer, 1999. 52

[147] G. H. Hardy and E. M. Wright. *An Introduction to the Theory of Numbers*. Oxford Science Publications, 2000. Fifth Edition.

[148] Johan Håstad. Some optimal inapproximability results. In *STOC*, pages 1–10, 1997.

[149] Johan Håstad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859, 2001. 128

[150] Wilko Henecka, Alexander May, and Alexander Meurer. Correcting errors in rsa private keys. In Tal Rabin, editor, *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference*, volume 6223 of *Lecture Notes in Computer Science*, pages 351–369. Springer, 2010. 19

[151] Nadia Heninger and Hovav Shacham. Reconstructing rsa private keys from random key bits. In Shai Halevi, editor, *Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference*, volume 5677 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 2009. 18

[152] Shoichi Hirose, Hidenori Kuwakado, and Hirotaka Yoshida. Sha-3 proposal: Lesamnta. Submission to NIST, 2008. 32

[153] Naofumi Homma, Atsushi Miyamoto, Takafumi Aoki, Akashi Satoh, and Adi Shamir. Comparative power analysis of modular exponentiation algorithms. *IEEE Trans. Computers*, 59(6):795–807, 2010.

[154] Nicholas J. Hopper and Manuel Blum. Secure human identification protocols. In Colin Boyd, editor, *Advances in Cryptology - ASIACRYPT 2001, 7th International Conference on the Theory and Application of Cryptology and Information Security*, volume 2248 of *Lecture Notes in Computer Science*, pages 52–66. Springer, 2001. 45, 127, 128

[155] Nick Howgrave-Graham. Finding small roots of univariate modular equations revisited. In Michael Darnell, editor, *Cryptography and Coding, 6th IMA International Conference*, volume 1355 of *Lecture Notes in Computer Science*, pages 131–142, 1997. 16

[156] Nick Howgrave-Graham and Antoine Joux. New generic algorithms for hard knapsacks. In Henri Gilbert, editor, *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, volume 6110 of *Lecture Notes in Computer Science*, pages 235–256. Springer, 2010. 14

[157] Nick Howgrave-Graham and Nigel P. Smart. Lattice attacks on digital signature schemes. *Des. Codes Cryptography*, 23(3):283–290, 2001.

[158] Russell Impagliazzo and Moni Naor. Efficient cryptographic schemes provably as secure as subset sum. *J. Cryptology*, 9(4):199–216, 1996. 11

[159] Kouichi Itoh, Tetsuya Izu, and Masahiko Takenaka. Address-bit differential power analysis of cryptographic schemes ok-ecdh and ok-ecdsa. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop*, volume 2523 of *Lecture Notes in Computer Science*, pages 129–143. Springer, 2003. 177

[160] Tetsuya Izu, Bodo Möller, and Tsuyoshi Takagi. Improved elliptic curve multiplication methods resistant against side channel attacks. In Alfred Menezes and Palash Sarkar, editors, *Progress in Cryptology - INDOCRYPT 2002, Third International Conference on Cryptology in India*, volume 2551 of *Lecture Notes in Computer Science*, pages 296–313. Springer, 2002. 177

[161] Thomas Jakobsen and Lars R. Knudsen. The interpolation attack on block ciphers. In Eli Biham, editor, *Fast Software Encryption, 4th International Workshop, FSE '97*, volume 1267 of *Lecture Notes in Computer Science*, pages 28–40. Springer, 1997.

[162] Svante Janson, Stefano Lonardi, and Wojciech Szpankowski. On average sequence complexity. *Theor. Comput. Sci.*, 326(1-3):213–227, 2004. 145

[163] Antoine Joux. Multicollisions in iterated hash functions. application to cascaded constructions. In Matthew K. Franklin, editor, *Advances in Cryptology - CRYPTO 2004, 24th Annual International Cryptology Conference*, volume 3152 of *Lecture Notes in Computer Science*, pages 306–316. Springer, 2004. 38, 137

[164] Antoine Joux. *Algorithmic Cryptanalysis*. CRC Press, 2009.

[165] Antoine Joux and Louis Granboulan. A practical attack against knapsack based hash functions (extended abstract). In *EUROCRYPT*, pages 58–66, 1994.

[166] Antoine Joux and Reynald Lercier. "chinese & match", an alternative to atkin's "match and sort" method used in the sea algorithm. *Math. Comput.*, 70(234):827–836, 2001. 14

[167] Antoine Joux, Gwenaëlle Martinet, and Frédéric Valette. Blockwise-adaptive attackers: Revisiting the (in)security of some provably secure encryption models: Cbc, gem, iacbc. In Moti Yung, editor, *Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference*, volume 2442 of *Lecture Notes in Computer Science*, pages 17–30. Springer, 2002. 37

[168] Antoine Joux and Thomas Peyrin. Hash functions and the (amplified) boomerang attack. In Alfred Menezes, editor, *Advances in Cryptology - CRYPTO 2007, 27th Annual International Cryptology Conference*, volume 4622 of *Lecture Notes in Computer Science*, pages 244–263. Springer, 2007. 49, 137

[169] Antoine Joux, Guillaume Poupard, and Jacques Stern. New attacks against standardized macs. In Thomas Johansson, editor, *Fast Software Encryption, 10th International Workshop, FSE 2003*, volume 2887 of *Lecture Notes in Computer Science*, pages 170–181. Springer, 2003.

[170] Antoine Joux and Jacques Stern. Lattice reduction: A toolbox for the cryptanalyst. *J. Cryptology*, 11(3):161–185, 1998. 10, 11

[171] Marc Joye and Sung-Ming Yen. The montgomery powering ladder. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop*, volume 2523 of *Lecture Notes in Computer Science*, pages 291–302. Springer, 2003. 60

[172] Ari Juels and Stephen A. Weis. Authenticating pervasive devices with human protocols. In Victor Shoup, editor, *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference*, volume 3621 of *Lecture Notes in Computer Science*, pages 293–308. Springer, 2005. 128, 129, 130

[173] David Kahn. *The Codebreakers: The Story of Secret Writing*. Scribner, 1996. 3, 4

[174] Jonathan Katz and Ji Sun Shin. Parallel and concurrent security of the hb and hb^+ protocols. In Serge Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, volume 4004 of *Lecture Notes in Computer Science*, pages 73–87. Springer, 2006. 128, 130

[175] Jonathan Katz and Moti Yung. Complete characterization of security notions for probabilistic private-key encryption. In *STOC*, pages 245–254, 2000. 38

[176] Michael J. Kearns. Efficient noise-tolerant learning from statistical queries. *J. ACM*, 45(6):983–1006, 1998. 128

[177] Michael J. Kearns and Umesh V. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, 1994. 44

[178] John Kelsey and Tadayoshi Kohno. Herding hash functions and the nostradamus attack. In Serge Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, volume 4004 of *Lecture Notes in Computer Science*, pages 183–200. Springer, 2006. 41, 42, 138, 139, 140, 146

[179] John Kelsey and Bruce Schneier. Second preimages on n-bit hash functions for much less than 2^n work. In Ronald Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, volume 3494 of *Lecture Notes in Computer Science*, pages 474–490. Springer, 2005. 39, 42, 137, 138

[180] Veikko Keränen. Abelian Squares are Avoidable on 4 Letters. In *ICALP*, pages 41–52, 1992. 141, 150

[181] Veikko Keränen. On abelian square-free DT0L-languages over 4 letters. In Tero Harju, editor, *WORDS'03*, volume 27, pages 95–109. TUCS General Publication, 2003. 141, 150

[182] Veikko Keränen. A powerful abelian square-free substitution over 4 letters. *Theor. Comput. Sci.*, 410(38-40):3893–3900, 2009. 42

[183] Joe Kilian and Phillip Rogaway. How to protect des against exhaustive key search. In Neal Koblitz, editor, *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference*, volume 1109 of *Lecture Notes in Computer Science*, pages 252–267. Springer, 1996. 37

[184] Jongsung Kim, Alex Biryukov, Bart Preneel, and Seokhie Hong. On the security of hmac and nmac based on haval, md4, md5, sha-0 and sha-1 (extended abstract). In Roberto De Prisco and Moti Yung, editors, *Security and Cryptography for Networks, 5th International Conference*, volume 4116 of *Lecture Notes in Computer Science*, pages 242–256. Springer, 2006. 49, 154, 155

[185] Aviad Kipnis, Jacques Patarin, and Louis Goubin. Unbalanced oil and vinegar signature schemes. In *EUROCRYPT*, pages 206–222, 1999. 63

[186] Aviad Kipnis and Adi Shamir. Cryptanalysis of the hfe public key cryptosystem by relinearization. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference*, volume 1666 of *Lecture Notes in Computer Science*, pages 19–30. Springer, 1999. 71, 73

[187] Vlastimil Klima. Tunnels in Hash Functions: MD5 Collisions Within a Minute. Cryptology ePrint Archive, Report 2006/105, 2006. `http://eprint.iacr.org/`. 137

[188] Vlastimil Klima. Tunnels in hash functions: Md5 collisions within a minute. Cryptology ePrint Archive, Report 2006/105, 2006. `http://eprint.iacr.org/`.

[189] Lars R. Knudsen. Truncated and higher order differentials. In Bart Preneel, editor, *Fast Software Encryption: Second International Workshop*, volume 1008 of *Lecture Notes in Computer Science*, pages 196–211. Springer, 1995. 33, 48

[190] Lars R. Knudsen. Smash - a cryptographic hash function. In Henri Gilbert and Helena Handschuh, editors, *Fast Software Encryption: 12th International Workshop, FSE 2005*, volume 3557 of *Lecture Notes in Computer Science*, pages 228–242. Springer, 2005. 14

[191] Lars R. Knudsen, Christian Rechberger, and Søren S. Thomsen. The grindahl hash functions. In Alex Biryukov, editor, *Fast Software Encryption, 14th International Workshop, FSE 2007*, volume 4593 of *Lecture Notes in Computer Science*, pages 39–57. Springer, 2007. 49

[192] Donald E. Knuth. *The Art of Computer Programming, Volume II: Seminumerical Algorithms, 2nd Edition*. Addison-Wesley, 1981. 185, 200

[193] Cetin K. Koç. High Speed RSA Implementation. Technical Report 201, RSA Labs, 1994. 200

[194] Cetin K. Koç. Analysis of sliding window technique for exponentiation. *Computers and Mathematics with Applications*, 10(30):17–24, 1995. 200

[195] Paul C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In Neal Koblitz, editor, *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer, 1996. 51, 173, 183, 184, 187, 199, 200

[196] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999. 55, 173, 175, 183

[197] Eyal Kushilevitz and Yishay Mansour. Learning decision trees using the fourier sprectrum (extended abstract). In *Proceedings of the Twenty Third Annual ACM Symposium on Theory of Computing*, pages 455–464. ACM, 1991. 48

[198] RSA Labs. PKCS #1 v2.1 : RSA Cryptography Standard. Technical report, RSA Labs, 2002. `http://www.rsalabs.com/pkcs/pkcs-1`. 184, 185

[199] J. C. Lagarias. Knapsack public key cryptosystems and diophantine approximation. In *CRYPTO*, pages 3–23, 1983. 20

[200] J. C. Lagarias. The computational complexity of simultaneous diophantine approximation problems. *SIAM J. Comput.*, 14(1):196–209, 1985. 21

[201] Xuejia Lai. Higher order derivatives and differential cryptanalysis. In *Communications and Cryptology*, pages 227–233. Kluwer Academic Publishers, 1994. 33

[202] Tri Van Le, Rüdiger Sparr, Ralph Wernsdorf, and Yvo Desmedt. Complementation-like and cyclic properties of aes round functions. In Hans Dobbertin, Vincent Rijmen, and Aleksandra Sowa, editors, *Advanced Encryption Standard - AES, 4th International Conference, AES 2004*, volume 3373 of *Lecture Notes in Computer Science*, pages 128–141. Springer, 2005. 33

[203] Gaëtan Leurent. Message freedom in md4 and md5 collisions: Application to apop. In Alex Biryukov, editor, *Fast Software Encryption, 14th International Workshop, FSE 2007*, volume 4593 of *Lecture Notes in Computer Science*, pages 309–328. Springer, 2007. 49, 163, 166

[204] Gaëtan Leurent, Charles Bouillaguet, and Pierre-Alain Fouque. Simd is a message digest. Submission to NIST, 2008. 38, 41, 49

[205] Éric Levieil and Pierre-Alain Fouque. An improved lpn algorithm. In Roberto De Prisco and Moti Yung, editors, *Security and Cryptography for Networks, 5th International Conference, SCN 2006*, volume 4116 of *Lecture Notes in Computer Science*, pages 348–359. Springer, 2006. 46, 127

[206] Shachar Lovett. Unconditional pseudorandom generators for low degree polynomials. In Cynthia Dwork, editor, *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, pages 557–562. ACM, 2008. 101

[207] Stefan Lucks. A failure-friendly design principle for hash functions. In Bimal K. Roy, editor, *Advances in Cryptology - ASIACRYPT 2005, 11th International Conference on the Theory and Application of Cryptology and Information Security*, volume 3788 of *Lecture Notes in Computer Science*, pages 474–494. Springer, 2005. 41

[208] Vadim Lyubashevsky. The parity problem in the presence of noise, decoding random linear codes, and the subset sum problem. In Chandra Chekuri, Klaus Jansen, José D. P. Rolim, and Luca Trevisan, editors, *Approximation, Randomization and Combinatorial Optimization, Algorithms and Techniques, 8th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2005 and 9th InternationalWorkshop on Randomization and Computation*, volume 3624 of *Lecture Notes in Computer Science*, pages 378–389. Springer, 2005. 46

[209] David J. C. MacKay. *Exact Marginalization in Graphs, Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003. 53

[210] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power Analysis Attacks - Revealing the Secrets of Smart Cards*. Springer-Verlag, 2007. 55

[211] Mitsuru Matsui. The first experimental cryptanalysis of the data encryption standard. In Yvo Desmedt, editor, *Advances in Cryptology - CRYPTO '94, 14th Annual International Cryptology Conference*, volume 839 of *Lecture Notes in Computer Science*, pages 1–11. Springer, 1994. 6, 32, 48

[212] Tsutomu Matsumoto and Hideki Imai. Public quadratic polynominal-tuples for efficient signature-verification and message-encryption. In *EUROCRYPT*, pages 419–453, 1988. 22, 28, 63, 71, 73, 81, 82

[213] Robert J. McEliece. A Public-Key Cryptosystem Based On Algebraic Coding Theory. Technical Report 42-44, DNS Progress Report, January-February 1978. 63

[214] Willi Meier and Othmar Staffelbach. Fast correlation attacks on certain stream ciphers. *J. Cryptology*, 1(3):159–176, 1989.

[215] A. J. Menezes, P. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996. 32, 37, 38, 60, 138, 185

[216] Ralph C. Merkle. One way hash functions and des. In Gilles Brassard, editor, *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference*, volume 435 of *Lecture Notes in Computer Science*, pages 428–446. Springer, 1990. 14, 38, 139

[217] Thomas S. Messerges, Ezzy A. Dabbish, and Robert H. Sloan. Power analysis attacks of modular exponentiation in smartcards. In Çetin Kaya Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems, First International Workshop, CHES'99*, volume 1717 of *Lecture Notes in Computer Science*, pages 144–157. Springer, 1999. 55, 174, 176, 200

[218] Lorenz Minder and Alistair Sinclair. The extended -tree algorithm. In Claire Mathieu, editor, *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2009*, pages 586–595. SIAM, 2009. 13

[219] Ilya Mironov. Hash functions: From merkle-damgård to shoup. In Birgit Pfitzmann, editor, *Advances in Cryptology - EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques*, volume 2045 of *Lecture Notes in Computer Science*, pages 166–181. Springer, 2001. 149

[220] M. Mitzenmacher and E. Upfal. *Probability and computing*. Cambridge University Press, 2006. 136

[221] Peter L. Montgomery. Speeding the pollard and elliptic curve methods of factorization. *Mathematics of Computation*, 48(177):243–264, 1987. 60

[222] François Morain and J. Olivos. Speeding up the computation on an elliptic curve using addition-substraction chains. *Inform Theory Appl.*, 24:531–543, 1990. 179

[223] David Naccache, Phong Q. Nguyen, Michael Tunstall, and Claire Whelan. Experimenting with faults, lattices and the dsa. In Serge Vaudenay, editor, *Public Key Cryptography - PKC 2005, 8th International Workshop on Theory and Practice in Public Key Cryptography*, volume 3386 of *Lecture Notes in Computer Science*, pages 16–28. Springer, 2005. 51

[224] Phong Nguyen. *Théorie et Pratique de la Cryptanalyse à Clef Publique*. PhD thesis, Université de Paris 7, Novembre 2007. 9

[225] Phong Q. Nguyen and Igor Shparlinski. The insecurity of the digital signature algorithm with partially known nonces. *J. Cryptology*, 15(3):151–176, 2002. 51

[226] Phong Q. Nguyen and Jacques Stern. The two faces of lattices in cryptology. In Joseph H. Silverman, editor, *Cryptography and Lattices, International Conference, CaLC 2001*, volume 2146 of *Lecture Notes in Computer Science*, pages 146–180. Springer, 2001. 10

[227] NIST. Fips pub 186-3: Digital signature standard (dss). `http://csrc.nist.gov/publications/fips/fips186-3/fips_186-3.pdf`, 2010. 57, 61, 173

[228] Roman Novak. Spa-based adaptive chosen-ciphertext attack on rsa implementation. In David Naccache and Pascal Paillier, editors, *Public Key Cryptography, 5th International Workshop on Practice and Theory in Public Key Cryptosystems*, volume 2274 of *Lecture Notes in Computer Science*, pages 252–262. Springer, 2002. 183, 184, 185, 187

[229] Kaisa Nyberg and Lars R. Knudsen. Provable security against a differential attack. *J. Cryptology*, 8(1):27–37, 1995. 32

[230] National Bureau of Standards. Data encryption standard, 1977. Federal Information Processing Standards Publications No.46. 31, 35, 48, 60

[231] Katsuyuki Okeya and Kouichi Sakurai. A second-order dpa attack breaks a window-method based countermeasure against side channel attacks. In Agnes Hui Chan and Virgil D. Gligor, editors, *Information Security, 5th International Conference, ISC 2002*, volume 2433 of *Lecture Notes in Computer Science*, pages 389–401. Springer, 2002. 176

[232] Eli Biham Orr Dunkelman. A Framework for Iterative Hash Functions — HAIFA. Presented at the second NIST hash workshop, 2006. 42, 138, 142, 144

[233] Dag Arne Osvik, Adi Shamir, and Eran Tromer. Cache attacks and countermeasures: The case of aes. In David Pointcheval, editor, *Topics in Cryptology - CT-RSA 2006, The Cryptographers' Track at the RSA Conference 2006*, volume 3860 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2006. 53

[234] Elisabeth Oswald and Manfred Josef Aigner. Randomized addition-subtraction chains as a countermeasure against power attacks. In Çetin Kaya Koç, David Naccache, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2001, Third International Workshop*, volume 2162 of *Lecture Notes in Computer Science*, pages 39–50. Springer, 2001.

[235] D. Page. Theoretical Use of Cache Memory as a Cryptanalytic Side-Channel. Technical Report CSTR-02-003, Department of computer science, university of Bristol, 2002. `http://www.cs.bris.ac.uk/Publications/Papers/1000625.pdf`. 53

[236] A. G. Pakes. Some limit theorems for the total progeny of a branching process. *Advances in Applied Probability*, 3(1):176–192, 1971. 110

[237] J.-J. Pansiot. Complexité des facteurs des mots infinis engendrés par morphismes itérés. In Jan Paredaens, editor, *11th ICALP, Antwerpen*, volume 172 of *LNCS*, pages 380–389. Springer, july 1984. 150

[238] Jacques Patarin. Cryptoanalysis of the matsumoto and imai public key scheme of eurocrypt'88. In Don Coppersmith, editor, *Advances in Cryptology - CRYPTO '95, 15th Annual International Cryptology Conference*, volume 963 of *Lecture Notes in Computer Science*, pages 248–261. Springer, 1995. 23, 25, 26, 29, 71, 73, 80, 81, 82, 83, 87, 89, 91

[239] Jacques Patarin. Asymmetric cryptography with a hidden monomial. In Neal Koblitz, editor, *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference*, volume 1109 of *Lecture Notes in Computer Science*, pages 45–60. Springer, 1996. 71, 72

[240] Jacques Patarin. Hidden fields equations (hfe) and isomorphisms of polynomials (ip): Two new families of asymmetric algorithms. In *EUROCRYPT*, pages 33–48, 1996. 22, 25, 63, 71, 90, 92, 93, 100

[241] Jacques Patarin, Nicolas Courtois, and Louis Goubin. Flash, a fast multivariate signature algorithm. In David Naccache, editor, *Topics in Cryptology - CT-RSA 2001, The Cryptographer's Track at RSA Conference 2001*, volume 2020 of *Lecture Notes in Computer Science*, pages 298–307. Springer, 2001. 22, 25, 26, 81, 83

[242] Jacques Patarin, Louis Goubin, and Nicolas Courtois. $^{*}_{-+}$ and hm: Variations around two schemes of t. matsumoto and h. imai. In Kazuo Ohta and Dingyi Pei, editors, *Advances in Cryptology - ASIACRYPT '98, International Conference on the Theory and Applications of Cryptology and Information Security*, volume 1514 of *Lecture Notes in Computer Science*, pages 35–49. Springer, 1998. 72, 73, 83

[243] Jacques Patarin, Louis Goubin, and Nicolas Courtois. Improved algorithms for isomorphisms of polynomials. In *EUROCRYPT*, pages 184–200, 1998. 30, 92, 93, 100

[244] Jacques Patarin and Valérie Nachef. "i shall love you up to the death" (marie-antoinette to axel von fersen). `http://eprint.iacr.org/2009/166.pdf`, 2009.

[245] Erez Petrank and Charles Rackoff. Cbc mac for real-time data sources. *J. Cryptology*, 13(3):315–338, 2000. 43

[246] J. M. Pollard. Monte carlo methods for index computation (mod p). *Mathematics of Computation*, 32(143):918–924, 1978. 11

[247] John M. Pollard. Kangaroos, monopoly and discrete logarithms. *J. Cryptology*, 13(4):437–447, 2000. 12

[248] Norbert Pramstaller, Christian Rechberger, and Vincent Rijmen. Breaking a new hash function design strategy called smash. In Bart Preneel and Stafford E. Tavares, editors, *Selected Areas in Cryptography, 12th International Workshop, SAC 2005*, volume 3897 of *Lecture Notes in Computer Science*, pages 233–244. Springer, 2006. 14

[249] Bart Preneel and Paul C. van Oorschot. Mdx-mac and building fast macs from hash functions. In Don Coppersmith, editor, *Advances in Cryptology - CRYPTO '95, 15th Annual International Cryptology Conference*, volume 963 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 1995. 155

[250] Bart Preneel and Paul C. van Oorschot. On the security of two mac algorithms. In *EUROCRYPT*, pages 19–32, 1996. 155

[251] Bart Preneel and Paul C. van Oorschot. On the security of iterated message authentication codes. *IEEE Transactions on Information Theory*, 45(1):188–199, 1999. 155, 156

[252] Christian Rechberger and Vincent Rijmen. Note on Distinguishing, Forgery, and Second Preimage Attacks on HMAC-SHA-1 and a Method to Reduce the Key Entropy of NMAC. Cryptology ePrint Archive, Report 2006/290, 2006. `http://eprint.iacr.org/`. 155

[253] Christian Rechberger and Vincent Rijmen. On authentication with hmac and non-random properties. In Sven Dietrich and Rachna Dhamija, editors, *Financial Cryptography and Data Security, 11th International Conference, FC 2007*, volume 4886 of *Lecture Notes in Computer Science*, pages 119–133. Springer, 2007. 155, 156, 164

[254] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, pages 84–93. ACM, 2005. 128, 130

[255] Ronald L. Rivest. Abelian Square-Free Dithering for Iterated Hash Functions. *Presented at ECrypt Hash Function Workshop, June 21, 2005, Cracow, and at the Cryptographic Hash workshop, November 1, 2005, Gaithersburg, Maryland*, August 2005. 138, 141, 150

[256] Ronald L. Rivest and Adi Shamir. Efficient factoring based on partial information. In *EUROCRYPT*, pages 31–34, 1985. 10

[257] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems (reprint). *Commun. ACM*, 26(1):96–99, 1983. 5, 81, 173, 174

[258] Phillip Rogaway and Thomas Shrimpton. Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In Bimal K. Roy and Willi Meier, editors, *Fast Software Encryption, 11th International Workshop, FSE 2004*, volume 3017 of *Lecture Notes in Computer Science*, pages 371–388. Springer, 2004. 38

[259] Pierre Samuel. *Théorie algébrique des nombres.* Hermann, 1967. 25

[260] Yu Sasaki, Lei Wang, Kazuo Ohta, and Noboru Kunihiro. New message difference for md4. In Alex Biryukov, editor, *Fast Software Encryption, 14th International Workshop, FSE 2007*, volume 4593 of *Lecture Notes in Computer Science*, pages 329–348. Springer, 2007. 49

[261] Werner Schindler. A timing attack against rsa with the chinese remainder theorem. In Çetin Kaya Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2000, Second International Workshop*, volume 1965 of *Lecture Notes in Computer Science*, pages 109–124. Springer, 2000. 183, 184

[262] W. M. Schmidt. On badly approximable numbers and certain games. *Trans. Amer. Math. Soc*, 123:178–199, 1966. 20

[263] Claus-Peter Schnorr. Efficient signature generation by smart cards. *J. Cryptology*, 4(3):161–174, 1991. 11

[264] Claus-Peter Schnorr. Factoring integers and computing discrete logarithms via diophantine approximations. In *EUROCRYPT*, pages 281–293, 1991. 20

[265] Claus-Peter Schnorr and M. Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Math. Program.*, 66:181–199, 1994. 9

[266] Kai Schramm, Thomas J. Wollinger, and Christof Paar. A new class of collision attacks and its application to des. In Thomas Johansson, editor, *Fast Software Encryption, 10th International Workshop, FSE 2003*, volume 2887 of *Lecture Notes in Computer Science*, pages 206–222. Springer, 2003. 55, 178

[267] Richard Schroeppel and Adi Shamir. A $t = o(2^{n/2})$, $s = o(2^{n/4})$ algorithm for certain np-complete problems. *SIAM J. Comput.*, 10(3):456–464, 1981.

[268] Adi Shamir. Efficient signature schemes based on birational permutations. In Douglas R. Stinson, editor, *Advances in Cryptology - CRYPTO '93, 13th Annual International Cryptology Conference*, volume 773 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 1994. 25, 81, 91

[269] Akihiro Shimizu and Shoji Miyaguchi. Fast data encipherment algorithm feal. In *EUROCRYPT*, pages 267–278, 1987. 48

[270] Victor Shoup. A composition theorem for universal one-way hash functions. In Bart Preneel, editor, *Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques*, volume 1807 of *Lecture Notes in Computer Science*, pages 445–452. Springer, 2000. 139, 147, 148

[271] Victor Shoup. Number Theory C++ Library (NTL) version 5.0b, 2003. `http://www.shoup.net/`. 189

[272] Victor Shoup. *A Computational Introduction to Number Theory and Algebra*. Cambridge University Press, 2008. 56

[273] Thomas Siegenthaler. Decrypting a class of stream ciphers using ciphertext only. *IEEE Trans. Computers*, 34(1):81–85, 1985. 6, 47

[274] Simon Singh. *Histoire des Codes Secrets*. Livre de Poche, 2001. 3, 4

[275] Douglas R. Stinson. Some baby-step giant-step algorithms for the low hamming weight discrete logarithm problem. *Math. Comput.*, 71(237):379–391, 2002. 199, 209

[276] Anne Tardy-Corfdir and Henri Gilbert. A known plaintext attack of feal-4 and feal-6. In Joan Feigenbaum, editor, *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference*, volume 576 of *Lecture Notes in Computer Science*, pages 172–181. Springer, 1992. 6, 48

[277] Elena Trichina and Antonio Bellezza. Implementation of elliptic curve cryptography with built-in counter measures against side channel attacks. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop*, volume 2523 of *Lecture Notes in Computer Science*, pages 98–113. Springer, 2003. 177

[278] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In Henri Gilbert, editor, *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, volume 6110 of *Lecture Notes in Computer Science*, pages 24–43. Springer, 2010.

[279] Paul C. van Oorschot and Michael J. Wiener. A known plaintext attack on two-key triple encryption. In *EUROCRYPT*, pages 318–325, 1990. 36

[280] Serge Vaudenay. On the need for multipermutations: Cryptanalysis of md4 and safer. In Bart Preneel, editor, *Fast Software Encryption: Second International Workshop*, volume 1008 of *Lecture Notes in Computer Science*, pages 286–297. Springer, 1995. 49

[281] Serge Vaudenay. Provable security for block ciphers by decorrelation. In Michel Morvan, Christoph Meinel, and Daniel Krob, editors, *STACS 98, 15th Annual Symposium on Theoretical Aspects of Computer Science*, volume 1373 of *Lecture Notes in Computer Science*, pages 249–275. Springer, 1998.

[282] Serge Vaudenay. Security flaws induced by cbc padding - applications to ssl, ipsec, wtls ... In Lars R. Knudsen, editor, *Advances in Cryptology - EUROCRYPT 2002, International Conference on the Theory and Applications of Cryptographic Techniques*, volume 2332 of *Lecture Notes in Computer Science*, pages 534–546. Springer, 2002.

[283] Serge Vaudenay. The period-learning problem - application to factoring. personal communication, 2010. 48

[284] David Wagner. The boomerang attack. In Lars R. Knudsen, editor, *Fast Software Encryption, 6th International Workshop, FSE '99*, volume 1636 of *Lecture Notes in Computer Science*, pages 156–170. Springer, 1999. 48

[285] David Wagner. A generalized birthday problem. In Moti Yung, editor, *Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference*, volume 2442 of *Lecture Notes in Computer Science*, pages 288–303. Springer, 2002. 12, 45, 130, 134

[286] Colin D. Walter. Sliding windows succumbs to big mac attack. In Çetin Kaya Koç, David Naccache, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2001, Third International Workshop*, volume 2162 of *Lecture Notes in Computer Science*, pages 286–299. Springer, 2001. 199

[287] Colin D. Walter. Seeing through mist given a small fraction of an rsa private key. In Marc Joye, editor, *Topics in Cryptology - CT-RSA 2003, The Cryptographers' Track at the RSA Conference 2003*, volume 2612 of *Lecture Notes in Computer Science*, pages 391–402. Springer, 2003. 199

[288] Lei Wang, Kazuo Ohta, and Noboru Kunihiro. New key-recovery attacks on hmac/nmac-md4 and nmac-md5. In Nigel P. Smart, editor, *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, volume 4965 of *Lecture Notes in Computer Science*, pages 237–253. Springer, 2008. 49

[289] Lei Wang, Yu Sasaki, Kazuo Sakiyama, and Kazuo Ohta. Bit-free collision: Application to apop attack. In Tsuyoshi Takagi and Masahiro Mambo, editors, *Advances in Information and Computer Security, 4th International Workshop on Security, IWSEC 2009*, volume 5824 of *Lecture Notes in Computer Science*, pages 3–21. Springer, 2009. 49

[290] Xiaoyun Wang, Xuejia Lai, Dengguo Feng, Hui Chen, and Xiuyuan Yu. Cryptanalysis of the hash functions md4 and ripemd. In Ronald Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, volume 3494 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2005. 49, 137, 153, 154, 157

[291] Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding collisions in the full sha-1. In Victor Shoup, editor, *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference*, volume 3621 of *Lecture Notes in Computer Science*, pages 17–36. Springer, 2005. 49, 137, 153, 154

[292] Xiaoyun Wang and Hongbo Yu. How to break md5 and other hash functions. In Ronald Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, volume 3494 of *Lecture Notes in Computer Science*, pages 19–35. Springer, 2005. 49, 137, 153, 154

[293] Xiaoyun Wang, Hongbo Yu, and Yiqun Lisa Yin. Efficient collision search attacks on sha-0. In Victor Shoup, editor, *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference*, volume 3621 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2005. 49, 137, 153, 154

[294] Michael C. Wendl. Collision probability between sets of random variables. *Statistics & Probability Letters*, 64(3):249 – 254, 2003. 108

[295] Michael J. Wiener. Cryptanalysis of short rsa secret exponents. *IEEE Transactions on Information Theory*, 36(3):553–558, 1990.

[296] Christopher Wolf and Bart Preneel. Equivalent keys in hfe, c^*, and variations. In Ed Dawson and Serge Vaudenay, editors, *Progress in Cryptology - Mycrypt 2005, First International Conference on Cryptology in Malaysia*, volume 3715 of *Lecture Notes in Computer Science*, pages 33–49. Springer, 2005. 94

[297] Dingfeng Ye, Kwok-Yan Lam, and Zong-Duo Dai. Cryptanalysis of "2 r" schemes. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference*, volume 1666 of *Lecture Notes in Computer Science*, pages 315–325. Springer, 1999. 72

[298] Sung-Ming Yen and Marc Joye. Checking before output may not be enough against fault-based cryptanalysis. *IEEE Trans. Computers*, 49(9):967–970, 2000.

[299] Hongbo Yu, Gaoli Wang, Guoyan Zhang, and Xiaoyun Wang. The second-preimage attack on md4. In Yvo Desmedt, Huaxiong Wang, Yi Mu, and Yongqing Li, editors, *Cryptology and Network Security, 4th International Conference, CANS 2005*, volume 3810 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2005. 50, 154, 157

Résumé

La cryptanalyse, anciennement l'*art de déchiffrer les messages secrets*, s'est rapidement développée depuis la Seconde Guerre mondiale. Durant cette guerre, le nombre total de messages chiffrés interceptés est estimé entre deux et trois millions. Pour accélérer les décryptements, les cryptanalystes britanniques ont mis au point le premier ordinateur, appelé Colossus. C'est ainsi que la cryptanalyse est devenue une science à l'intersection des mathématiques et de l'informatique.

Cette thèse décrit plusieurs classes d'attaques cryptographiques qui diffèrent par les techniques employées. Les premiers outils mathématiques utilisés en cryptanalyse sont les statistiques : elles ont permis d'exploiter les distributions de probabilité non-uniformes de répartition des lettres dans les langues naturelles et, plus tard, les distributions biaisées des sorties des mécanismes cryptographiques. Puis, après l'invention de la cryptographie à clé publique, les cryptanalystes ont fait appel à l'algèbre pour étudier les schémas reliés à des problèmes difficiles en théorie de la complexité : le problème RSA, du logarithme discret, du sac-à-dos, ceux issus de la théorie des codes correcteurs d'erreurs, ou ceux concernant la résolution de systèmes polynomiaux en plusieurs variables dans un corps fini. Il est également possible en cryptanalyse de schémas symétriques d'utiliser des outils algébriques pour l'étudier des primitives, mais la mise en forme des équations à résoudre est souvent plus complexe qu'en cryptographie asymétrique. Enfin, les attaques par canaux auxiliaires constituent un nouvel outil pour les cryptanalystes, qui tirent ainsi profit des implémentations logicielles ou matérielles des systèmes cryptographiques. Ces implémentations laissent fuir des informations sur les variables internes manipulées au cours des calculs. La connaissance de ces informations, jusqu'alors non prises en compte, autorise un grand nombre d'attaques nouvelles très efficaces.

Mots-clés : Cryptanalyse, cryptographie à clé publique, cryptographie à clé secrète, attaque par canaux auxiliaires.