Cryptographie asymétrique et l'isomorphisme des polynômes (IP).

Nicolas COURTOIS
*
Magistère de Mathématiques Fondamentales,
Appliquées et d'Informatique (MMFAI),
Université Pierre et Marie Curie (Paris 6),
Bull CP 8,
LI/MS - Université de Toulon et du Var.
*

Sous direction de Jacques Patarin
Novembre 1997

Table des matières

Première partie

Mon magistère.

0.1 Préface

Le monde des mathématiques, dans lequel je suis rentré très tôt dans ma vie, semble calme et imperturbable. On a l'impression d'y apprendre des vérités éternelles, qui ne changent pas.

Je suis venu dans le magistère avec cette vision des mathématiques. Je me suis alors intéressé à la logique, pour découvrir dans toutes les théories scientifiques une double changeabilité.

D'une part, les axiomes de ces théories peuvent être profondément modifiés, d'autre part notre perception de ces axiomes peut évoluer (ce qui constitue en fait une couche de méta-axiomes).

Grâce au cursus interdisciplinaire du magistère j'ai pu m'orienter vers l'informatique, un domaine jeune, en mouvement, qui cherche ses bases.

La cryptographie, en particulier, est probablement le domaine de l'Informatique le plus instable, car c'est celui où les théories sont confrontées sans répit à l'épreuve de la réalité.

Ainsi j'avais évolué d'un monde plein de certitudes vers un monde de plus en plus incertain, complexe, et en constante évolution.

Le présent rapport est une continuation de mon travail de DEA qui portait sur les cryptosystèmes asymétriques appelés à Représentation Obscure (OR), définies en 4.3. Ces cryptosystèmes consistent à cacher la structure algébrique d'une fonction par des applications affines.

Dans mon rapport de DEA j'avais étudié des nombreux cryptosystèmes de ce type et obtenu un certain nombre de résultats dont le résumé se trouve en 1.2 et 1.5.

Grâce à Jacques Patarin mon attention avait été attirée par le problème de récupérer la clé secrète, sous-jacent à tous ces cryptosystèmes, qui est appelé IP (voir définition précise en 4.1.)

J'ai trouvé de nouvelles méthodes pour résoudre ce problème beaucoup plus facilement que celles connues avant, voir [19].

Ce problème a une portée beaucoup plus grande que la cryptographie asymétrique dont il est issu. Dans sa version généralisée MP, voir chapitre 12, il permet d'espérer trouver de nouveaux algorithmes, en particulier pour la multiplication rapide des matrices.

Pour cela il faudrait pouvoir généraliser les nouvelles méthodes trouvées pour le problème IP (chapitre 11, [19]) au problème MP. La manière d'y parvenir n'est pas encore claire.

0.2 Mon parcours.

1997 Stage en cryptographie chez Bull CP8, sous direction de Jacques PATARIN, note obtenue 15.

1996-97 DEA ALgorithmique, filière complexité, codage, cryptographie. Université Paris 6, note obtenue 13,55.

1995-96 L'année sabbatique.

1995 Admis 1-er au concours d'entrée en 3-ème année d'Informatique de l'École Normale Supérieure de Cachan, démissionné.

1994-95 Magistère de Mathématiques Fondamentales, Appliquées et d'Informatique (MMFAI) deuxième année.

1994-95 DEA de Logique et Fondements d'Informatique, Université Paris 7.

1994-95 Stage en cryptographie à l'Université de Toulon et du Var sous direction de Sami HARARI, note obtenue 16.

1995 Un article sur les méthodes heuristiques présenté au VI-ème Congrès de Philosophes Polonais à Toruǹ.

1994 Maîtrise des Matématiques, mention Mathématiques Pures, Univ. Paris 7.

1993-94 Magistère de Mathématiques Fondamentales, Appliquées et d'Informatique première année.

1991-1993 DEUG et Licence de Mathématiques, Université Paris 7.

1991 Institut Polytechnique de Gdaǹsk, Pologne - École d'Ingénieurs, Électronique.

1990 Olympiade Mathématique Internationale Polono-Autrichienne à Poznaǹ, 2- ème prix.

1990 Baccalauréat au Lycée M.Kopernik à Gdaǹsk, Pologne.

1987-90 Lauréat de nombreux concours : Mathématiques, Physique, Astronomie et Technique.

Deuxième partie

Le travail de DEA et sa suite.

Chapitre 1

Le contenu du rapport.

1.1 Remerciements

J'aimerais avant tout remercier Jacques PATARIN de m'avoir accueilli et encadré tout au long de ce stage, ainsi que pour tout ce que j'ai pu apprendre. Je remercie aussi chaleureusement Louis GOUBIN pour sa compagnie, son aide et ses conseils.
Je remercie Jacques STERN, Jean-Jacques QUISQUATER, Gilles BRASSARD et Sami HARARI de m'avoir enseigné la cryptologie.
Je remercie mes enseignants de DEA qui m'ont beaucoup apporté : Robert CORI, Pascale CHARPIN, Philippe FLAJOLET, François MORAIN, Miklos SANTHA, Jean-Marc STEYAERT, et d'autres.
Je remercie tous le personnel de Bull CP8 pour leur chaleureux accueil et une aide efficace pour toute chose.
Je remercie Isabelle GUERIN-LASSOUS de m'avoir volontairement fait profiter de son expérience.
Et tous ceux que j'aurais involontairement oubliés.

À la mémoire des mathématiciens victimes
de l'extermination planifiée, nazie et soviétique, des élites polonaises.

1.2 Résumé

La carte à puce est depuis 10 ans moteur d'innovation en cryptographie à clef publique. Dans Eurocrypt'88, C^*, un nouveau cryptosystème à clef publique, basé sur des polynômes multivariables de degré 2 est proposé par MATSUMOTO et IMAI. S'il avait été solide, il aurait été idéal pour la carte à puce. Cependant, il est cryptanalysé par Jacques PATARIN dans Crypto'95. Dans ce rapport, nous étudions la sécurité des systèmes similaires, dans le contexte précis des applications dans des cartes à puce.

Nous montrons une attaque surprenante sur un système appelé D^*, cryptanalysé depuis peu par une attaque indépendante. Notre méthode permet même d'obtenir la clef secrète. On ne savait le faire jusqu'ici pour aucun cryptosystème de la famille. Ce problème est en toute généralité appelé IP. Nous présenterons quatre nouvelles méthodes pour le résoudre, dont la meilleure est en $\mathcal{O}(q^{n/2})$. Le meilleure attaque connue jusque là était en $\mathcal{O}(q^{n\sqrt{n}})$. Cela permet d'espérer de trouver de nouveaux algorithmes pour la multiplication rapide de matrices.

Nous montrons qu'un autre système HM, basé sur les matrices, proposé en 1985 n'est pas sûr, et nous étudions comment le réparer. Nous montrons aussi un certain nombre de propriétés génériques des cryptosystèmes étudiés qui mènent à des nouvelles attaques. En particulier, nous montrons comment attaquer "Scotch". Nous étudions avec soin des conditions d'utilisation des boî tes S dans des nouveaux cryptosystèmes.

1.3 Abstract

Smart card implementations has been, for the last 10 years, a major inducement for innovation in public key cryptography. In EUROCRYPT'88 , the C^*, a new, multivariate quadratic equation based, public key cryptosystem is proposed by MATSUMOTO and IMAI. Strikingly simple, it requires amazingly little computation. Jacques Patarin broke it in Crypto'95. In this work we have been studying similar ideas within the framework of possible smart card applications.

We show a surprising attack of a cryptosystem called D^*, recently known to be independently broken. Yet our approach seems more powerful and allows us to find the whole secret key, which hasn't been done for any cryptosystem of the kind. This an exemple of a general problem called IP. We introduce four new solving techniques, the best in $\mathcal{O}(q^{n/2})$. The best complexity so far was $\mathcal{O}(q^{n\sqrt{n}})$. It gives hope to find new better fast matrix multiplication algorithms.

We also show that another matrices-based system HM, proposed in 1985, is not secure and we inquire how to repair it. We point out several generic properties of all studied schemes which suggest new attacks. We show how to attack "Scotch" and we have determined precise conditions of use of S- boxes in such schemes and their security.

1.4 L'origine des problèmes étudiés.

J'avais écrit l'introduction qui suit sur le site Bull de Louveciennes, l'ancien emplacement du Quartier Général de l'Alliance Atlantique, au moment même où la France a fait des pas pour resserrer ses liens avec ce pacte, et également quand mon pays natal, la Pologne, l'éternel allié de la France, a commencé les négociations d'adhésion. C'est ici que j'ai eu la chance de faire mon stage de D.E.A. dans le domaine fascinant de la cryptologie, sous direction de Jacques Patarin.

Nous approchons aujourd'hui d'une nouvelle période de notre histoire que l'on pourrait appeler l'ère de l'information. Non seulement l'importance de cette information dans notre société ne cesse de croî tre, mais elle tend manifestement à devenir sa principale ressource et richesse, en se substituant à l'argent. De plus en plus, d'autres richesses en découlent, ou lui sont subordonnées. La cryptologie, qui par définition sert à protéger cette information contre tout abus, et à prévenir des risques et dangers des nouvelles technologies (d'information), prend une importance capitale.

Jadis, elle jouait un rôle prépondérant dans la diplomatie et la guerre. Nous lui devons de beaux chapitres de la coopération internationale, en particulier entre la Pologne et ses alliés, avec la cryptanalyse d'Enigma par des jeunes cryptologues polonais dans les années trente, qui a joué un rôle essentiel dans la deuxième guerre mondiale.

Le temps des guerres est révolu. La cryptologie moderne n'a pourtant rien d'une arme obsolète, et au contraire, devient un art martial noble et à haute utilité publique. Elle sert avant tout la cause de la paix, de la justice, de l'ordre public et de la démocratie, et connaî t un succès grandissant. La bataille moderne est celle de l'homme contre ses propres faiblesses, et plus celle des uns contre les autres. La fraternité des hommes au dessus des nations n'est plus dans le champ de bataille, et fort heureusement, mais elle s'exprime surtout dans le travail commun, en particulier dans les entreprises multinationales comme Bull.

Avec la carte à puce (en anglais "smart card"), Bull CP8 œuvre activement à développer et à diffuser dans le monde entier les technologies de pointe dans le domaine de cryptologie. On compte des nombreuses applications dans la carte bancaire, le portefeuille électronique, les cartes d'accès, le dossier médical, le parc-mètre, la télévision à péage, ou encore dans la téléphonie. Et dans tous les cas, il n'est pas uniquement question de la sécurité des transactions financières, des banques, des entreprises ou d'organismes publics, mais également de la protection des libertés individuelles de chacun, d'assurer un meilleur service et qualité de vie, d'un surcroî t de liberté et de sécurité, et tout cela à moindre coût. Il est certain que le développement de la carte à puce n'a pas encore atteint son apogée, et que son rôle, dans les années à venir, ne cessera de croî tre.

Avant toutefois que la carte ne devienne un produit de première nécessité, il est important de consacrer d'importants moyens de recherche à assurer sa sécurité.
Les cryptosystèmes actuellement utilisés dans une carte à puce (p.ex. DES, RSA ou

Guillou-Quisquater) sont très peu nombreux. Or, il peut paraî tre inquiétant que la sécurité du "monde libre" dépende seulement de quelques 2 ou 3 problèmes mathématiques. Cela motive la recherche de nouveaux systèmes.

L'état actuel de l'art dans le domaine des algorithmes asymétriques, donne des solutions à sécurité éprouvée, comme déjà mentionné le RSA, mais peu efficaces en implémentation ou trop gourmandes en mémoire.
Il s'y ajoutent également des problèmes de brevets.

Il devient alors impossible, à l'heure actuelle, de profiter pleinement de ces systèmes dans les cartes à puce qui doivent rester peu chères à produire.
Contrairement à ce qu'on a habitude de voir dans les microprocesseurs actuels, qui contiennent des millions de transistors, une carte à puce, à cause de petites dimensions des puces, et de leur fragilité ne dispose que d'extrêmement peu.
On cherche donc des nouveaux outils cryptographiques, où les calculs puissent être effectués très facilement et avec très peu de mémoire.

Telles sont les principales raisons qui ont motivé cette recherche de nouvelles voies dans le domaine de la cryptographie asymétrique.

1.5 Guide de lecture pour le rapport.

Dans le chapitre suivant on présente l'entreprise Bull et sa filiale CP8. Ensuite, le passé et le présent de la carte à puce sont retracés. Dans le chapitre 4 on présente les cryptosystèmes qui ont été étudiés, et dans le chapitre 5 on fait le point de leur propriétés communes.

Nous avons étudié plus particulièrement 5 cryptosystèmes de la famille, et à chacun d'entre eux nous consacrons un chapitre entier.

Ainsi, dans le chapitre 6 on étudie les systèmes contenant des boî tes S aléatoires, et on en fixe des paramètres critiques à partir desquels on ne sait plus cryptanalyser les schémas à deux étages qui leurs sont associés. Dans le chapitre 7 on parlera d'un système basé sur la multiplication matricielle appelé HM, de sa cryptanalyse et l'on verra comment le réparer.

Le chapitre 8 expose une surprenante attaque du cryptosystème D^*, et encore plus surprenant, on verra comment récupérer la totalité de la clef secrète. C'est problème s'appelle en toute généralité IP et D^* est le seul cas connu à ce jour où on résout le problème IP en temps polynomial.

Dans le chapitre 9 on expose quelques résultats de simulations concernant un autre

système , le HFE. Le cryptosystème HFE possède un plus par rapport à tous les autres, car la fonction interne du système possède un grand degré de liberté, et de ce fait elle peut être considérée comme secrète, ainsi il ne suffit pas de résoudre le problème IP pour le casser.

Enfin le chapitre 10 est consacré à une attaque récente du dernier système cryptanalysé, "Scotch".

Le chapitre 11 s'occupe du problème IP, étudie en détail sa complexité et présente quatre nouvelles méthodes d'attaque. Ces méthodes nous ont permis de réduire la complexité du problème IP de façon spectaculaire : $\mathcal{O}(q^{n/2})$ au lieu de $\mathcal{O}(q^{n\sqrt{n}})$.

Le problème IP se généralise naturellement en MP, le morphisme des polynômes décrit dans le chapitre 12. Ce problème est très intéressant car il permet d'espérer d'améliorer les algorithmes existants pour la multiplication rapide des matrices et bien d'autres algorithmes.

Dans le dernier chapitre (13), nous résumons ce que nous avons appris, découvert et obtenu expérimentalement et nous examinons des perspectives d'avenir pour le sujet.

Troisième partie

Bull, CP8 et cartes à puces.

Chapitre 2

Présentation de l'entreprise

2.1 La multinationale Bull.

Bull est une grande entreprise multinationale, qui se définit comme fournisseur et intégrateur des produits, systèmes et services informatiques pour les entreprises et administrations. Son chiffre d'affaires pour 1996 est de 24 milliards, dont 60% constituent les produits et systèmes, et 40% les services. Les actionnaires principaux de Bull sont : l'état français, qui après la récente privatisation a vu sa participation chuter de 37 à 17 %, aux côtés de France Télécom, Motorola et NEC, dont la participation est également de 17 % chacun. L'entreprise, sous la présidence de Jean-Marie Descarpentries, a renoué depuis quelques années avec les bénéfices, et son image de marque s'est nettement améliorée après des années 1990-94 catastrophiques. Le résultat net de l'entreprise pour 1996 est de 376 millions de francs.
Sa privatisation fut un grand succès. L'offre fut littéralement aspirée par la demande, plus de 20 fois supérieure et le prix des actions grimpa rapidement. Vendues à 11 F (première tranche de privatisation) et à 22 F (deuxième tranche) elles ont atteint à 61F à la fin juillet.

L'entreprise Bull emploie plus de 21 000 employés, dont 88% en Europe et 49% seulement en France. Cela implique que sa stratégie est avant tout européenne.

2.2 Bull CP8.

Bull CP8 anciennement CP8 Transac, est une filiale de Bull. Il fait partie de la division Bull PTS (Personal Transaction Systems). Elle se place actuellement au premier rang mondial dans le domaine de technologies de la carte à microprocesseur. Son chiffre d'affaires connaît une progression fulgurante. En 1996 il avait atteint le montant d'un milliard 72 millions francs, ce qui constitue 61 % de croissance nette par rapport à 1995. Bull PTS emploie 440 personnes.

2.3 Clients de Bull CP8

Le deux principaux clients de Bull CP8 sont les banques françaises et France Télécom. Des nombreuses banques étrangères ont également adopté la carte CP8, dont la Bank of America, la ROYAL BANK au Canada, les banques norvégiennes ou BANKSYS en Belgique qui avait acheté le porte-monnaie électronique. Un contrat important vient d'être signé en Hollande. On l'utilise aussi dans différents pays du monde pour le paiement dans les hôtels, les universités ou restaurants universitaires, et bien entendu pour le contrôle d'accès. Bull CP8 est également premier fournisseur en Europe et le troisième dans le monde, de lecteurs de cartes. Le marché mondial de la carte à puce est aujourd'hui de l'ordre de 300 millions de cartes par an (!) dont environ 65 millions sont des cartes à microprocesseur (le domaine de CP8), le reste consistant en des cartes plus simples, comme la télécarte, qui à terme seront sans doute remplacées par des cartes CP8. En l'an 2000, et on prévoit qu'en Europe seulement, 500 millions de cartes à microprocesseur seront vendues, et environ 1 milliard dans le monde.

2.4 Bull, la France et le monde

La France a été pionnière dans le domaine des applications de la carte à puce, et reste encore aujourd'hui, et de loin, le premier pays au monde au niveau du nombre de cartes. C'est d'autant plus significatif, qu'en général, dans le domaine des technologies d'information, nous avons pris un retard considérable (3 fois moins d'ordinateurs dans les foyers français comparé au niveau d'équipement américain, et 10 fois moins de connexions internet). En revanche, le marché américain n'a pas encore largement adopté la carte à puce. Il faut espérer, que les barrières culturelles vont être franchies, et qu'il y aura un rééquilibrage pour le profit de tous.

Chapitre 3

La carte à puce.

3.1 Histoire

L'histoire de la carte à puce est riche et mouvementée. On dit qu'elle remonte à "La nuit des temps", titre d'un roman de science-fiction du français René BARJAVEL (vers 1960), où il décrit un gadget éléctronique sous forme d'un bijou portatif, capable entre autres d'ouvrir des portes à son propriétaire. L'idée d'utiliser un composant électronique contenu dans une carte de crédit a commencé à faire son chemin dès 1967 donnant lieu à des nombreux brevets en Europe, aux États Unis et au Japon. Le plus souvent ces projets n'ont pas donné lieu à des réalisations industrielles, car ils anticipaient les technologies disponibles.

Celui que la presse et la télévision considèrent comme l'inventeur de la carte à puce, Roland MORENO, a déposé entre 1974 et 1975 six brevets qui ont donné lieu, entre autres, à la télécarte française en circulation depuis 1983.
Roland MORENO est un bel exemple de réussite financière d'un inventeur, avec un total de 500 millions de francs de royalties, mais il faut se rappeler que ce n'est pas l'idée de la carte à puce elle-même qui compte le plus. Ce qui importe avant tout, c'est la façon dont elle avait été décrite, élargissant au maximum le champ de l'application du brevet, et en divisant en plusieurs brevets distincts, ainsi que le moment opportun de déposer le brevet (leur durée est limitée).

Certains n'ont pas eu de chance, comme l'américain J. ELLINGBOE qui avait écrit dans son brevet que la carte devrait avoir un côté biseauté, avec l'idée d'éviter de l'introduire à l'envers. Or aucune carte existante n'a jamais eu de côté biseauté, et le brevet n'a jamais rien rapporté.

La carte de Roland MORENO était une simple CARTE À MÉMOIRE, dite aussi CARTE À LOGIQUE CÂBLÉE (ang. **wired logic card**) et n'était pas programmable. Puis dès 1977, Michel UGON, à qui on avait confié l'étude du problème chez CII-Honeywell Bull se rendit compte que seule la présence d'un microprocesseur peut donner à la carte

suffisamment de fonctionnalités, surtout au niveau de la sécurité (algorithmes cryptographiques). Il devient clair que la carte doit être intelligente. Ainsi naquit le brevet et les réalisations pratiques qui donnèrent naissance, entre autre, à la carte à puce bancaire.

Il s'agit d'une carte dotée d'un microprocesseur, appelée aussi LA CARTE À MICRO-CALCULATEUR (ang. **IC card**, **smart card**) dont la première réalisation est appelée CP8. C'était une carte bi-puces, ce qui était un défaut considérable au niveau de la sécurité, car on pouvait imaginer de pouvoir se connecter sur des fils reliant la mémoire au microprocesseur. Pour cette raison, en 1981, en collaboration avec MOTOROLA, vit le jour le premier calculateur monolithique pour la carte à puces, appelé SPOM (**Self Programmable One Chip Computer**). Après des tests et les appels d'offre, le GIE (Groupement d'intérêt économique) Carte Bancaire représentant l'ensemble des banques françaises, commanda en 1985 à Bull, 16 millions de cartes à puce de type CP8, qui l'ont emporté sur tous les concurrents.

Jusqu'en 1992 les cartes et les lecteurs n'avaient pas encore la fiabilité d'aujourd'hui et n'était pas normalisés. C'est à partir de 1992 que la carte se généralisa avec un succès étonnant. Il suffit de dire qu'entre 1991 et 1994 la fraude par carte bancaire à été divisée par trois.

La carte à mémoire et la carte à microprocesseur coexistent aujourd'hui, la première présente l'avantage du prix, la deuxième plus de souplesse et de sécurité.
Il existe aussi une "super smart card", inventée par les japonais CASIO et TOSHIBA, qui contient un écran, un clavier, les contacts électriques et une bobine qui simule la piste magnétique. Cette invention n'a pas trouvé d'applications à cause de sa grande fragilité et de son prix élevé.
Il existe également des cartes qui fonctionnent sans contact (utilisées à la RATP) qui peuvent encore se généraliser.
La carte CP8 est largement utilisée. Toutefois, 22 % seulement des cartes vendues dans le monde sont des cartes de paiement. D'autres trouvent leur application dans les domaines de la santé, de la téléphonie, du transport, de la télévision à péage, etc. On a récemment décidé de l'adoption générale, en France, de la carte d'assuré social électronique. Le marché avait été emporté par Bull et Schlumberger.

Le concept du porte-monnaie électronique (P.M.E), largement adopté en Belgique, et ponctuellement dans de très nombreux pays, a connu un grand succès, avec entre 15 à 20 millions de cartes P.M.E. en circulation prévues en 1997.

3.2 Les fonctions d'une carte à puce

En total, Bull CP8 détient plus de 1200 brevets concernant la carte à puce et son environnement. Le domaine est complexe. D'une part, la même carte peut servir dans différentes applications (p.ex. celles mentionnées plus haut). D'autre part, à l'intérieur d'une application concrète, elle remplit des nombreuses fonctions. Ce sont principalement :

♣ **Authentification** La carte prouve au terminal son authenticité.

♣ **Identification** Son porteur est également authentifié par un mot de passe (PIN) que la carte vérifie.

♣ **Chiffrement** La carte peut chiffrer et déchiffrer des messages ou des données.

♣ **La signature électronique** Permet d'attester la provenance et l'intégrité d'un message. Selon l'application, la carte peut générer ou vérifier des signatures.

♣ **Certification** La carte certifie qu'une opération avait été effectuée, ou qu'une information se trouve dans la carte, pour cela un certificat est généré.

♣ **Dossier portable** La carte peut stocker des informations. Cette fonction, particulièrement intéressante pour les applications de dossier médical, n'est pas pour le moment très développée à cause de la faible quantité de mémoire contenue dans les cartes actuelles.

3.3 Réalisation pratique.

Pour remplir toutes ces fonctions, la puce est un véritable micro-ordinateur qui possède son propre système d'exploitation, un programme, une mémoire pour stocker des données, une interface entrée-sortie etc. Il s'agit en fait d'un micro contrôleur monolithique standard, de type microprocesseur de la famille Motorola 6805 ou autre, avec mémoire incorporée, enrichi des nombreuses fonction de sécurité.

Logique de sécurité d'une carte à puce. La composant est enrobé (passivation) de façon à interdire le retour de la puce en mode test et à prévenir toute sorte d'intrusion. La logique de sécurité contrôle l'alimentation, la fréquence d'horloge, mesure la résistance de l'enduit et contrôle la présence de la lumière. Il existe aussi des mesures de "design interne", visant à rendre le schéma incompréhensible (interdire le "reverse engineering"). On peut également ajouter des bits de contrôle dans la mémoire pour rendre difficile une modification locale, où encore utiliser un décodeur d'adresses spécial

de sorte a ce que les bits consécutifs ne soient pas consécutifs physiquement. La carte possède un système de fichiers hiérarchique complexe avec contrôle d'accès.

La fabrication des cartes. Le format standard de carte de crédit, avec son épaisseur de $3/4$ mm impose une contrainte considérable. Le processus d'introduction de la puce dans la carte appelé **encartage** est critique. D'abord l'épaisseur de la puce est de 0.5 mm seulement, la carte contient une piste magnétique et doit rester parfaitement lisse. Ensuite, elle est soumise couramment à des très sévères conditions d'exploitation. Ces opérations ont nécessité plusieurs années de mise au point pour devenir fiables et sont actuellement effectuées par des chaî nes robotisées.

Un autre problème, c'est la surface de la puce. Elle est entre 1 et 25 mm^2. Plus la surface est grande, plus on peut avoir de transistors, mais la puce risque alors de se briser ou encore de se décoller quand on plie légèrement la carte.

Au dessus de la puce il y a un *module électronique* avec des contacts, dont 5 sont utilisées à ce jour (du moins dans les interfaces normalisées) : GRD (terre), CLK (horloge), $+$5V, RAZ et I/O (port de communication). Les puces, découpées dans une galette de silicium après tout le processus de fabrication (nombreuses séances de photolithogravure etc..), sont découpées avec des scies diamantées, puis fixées au dos des modules (ang. "die bonding"). Ensuite, chacune des puces est connectée au module, principalement par la technique appelée "wire bonding", qui utilise de microscopiques fils d'or (20 μm de diamètre), soudés par thermocompression ou par ultrasons. C'est alors qu'on enrobe la puce dans la résine et qu'on colle l'ensemble dans la carte (encartage).

Un aspect technologique important extérieur à la carte, est celui des connecteurs. Les connecteurs bas de gamme sont "frottants", ont une action néfaste sur la carte (abrasion) et ne dépassent pas 10 000 utilisations. Des connecteurs haut de gamme sont de type "atterrissants", se posent sur la carte et peuvent dépasser plusieurs centaines de milliers de manœuvres.

L'électronique de la puce. Les puces sont réalisées en technologie CMOS, qui offre des nombreux avantages par rapport à la technologie bipolaire, comme les faibles dimensions et consommation d'énergie, l'immunité au bruit, ou la possibilité de réaliser des mémoires mortes programmables de faibles dimensions (et illisibles optiquement ou par des micro-intrusions), grâce à un haut niveau d'isolement.

La carte contient 3 types de mémoire :

ROM (mémoire morte). Elle contient le programme, et est réalisée par le masquage ou l'implantation ionique dans le silicium. Les premières cartes à puce disposaient

de 2 ko de ROM, aujourd'hui c'est environ 8 ko pour une carte bon marché, et jusqu'à 20 ko.

RAM (mémoire vive). C'est une mémoire utilisée pendant les calculs pour stocker les données intermédiaires. Dans les cartes à puce on utilise les mémoires de type SRAM (RAM statiques), qui ne nécessitent pas d'être périodiquement refraî chies comme les RAM dynamiques (DRAM), mais dont la cellule prend plus de place (4 à 6 transistors, comparé à 1 seul pour DRAM). Il en résulte un coût important. Une telle cellule occupe environ 20 fois plus de place qu'une cellule ROM. C'est la mémoire RAM qui coûte comparativement le plus cher dans une carte à puce, et la première carte française se contentait de 34 octets ! Actuellement elle est de l'ordre de 128 bits, et plus ($<$1 ko) dans les cartes plus chères.

E^{2}PROM (mémoire non-volatile réinscriptible). Contient des données de personnalisation et stocke des traces des opérations. La taille de cette mémoire dans les cartes actuelles varie entre 2 et 10 ko, typiquement 4 ko.
Les cartes sont alimentées à 5 V et cadencées à 3.579545 MHz (fréquence externe). Il y a un multiplieur interne de tension pour l'effacement des E^{2}PROM (12 à 20 V) et un démultiplieur de la fréquence d'horloge. Le taux de transfert de données entre la carte et le lecteur est de 9600 bits par seconde (seulement !), la communication est bidirectionnelle, en série sur 1 seul fil, et avec des signaux de contrôle, ce qui diminue encore le taux réel. Dès la connection, la carte réçoit de la part du terminal un signal RAZ, et envoie une *réponse à remise à zéro*, qui est un message standard contenant des informations concernant le type de la carte, le mode de transmission des caractères, la fréquence d'horloge, la vitesse de transmission, le protocole supporté, et d'autres caractéristiques propres.

3.3.1 Normalisation et systèmes d'exploitation.

Des nombreuses normes régulent le monde des cartes, en particulier la norme ISO 7816-1 et 7816-2. Deux emplacements standards y sont définis pour le contact de la carte, appelé BOUTON, les positions dites haute et basse.
La norme ISO-7816 définit la structure des fichiers dans une carte à puce, les attributs de sécurité, la messagerie de sécurité et le jeu de commandes du système d'exploitation. 3 classes de systèmes d'exploitation (OS) existent à l'heure actuelle :

Systèmes monoprestataires pour les cartes utilisant la cryptographie symétrique.

Systèmes multiprestataires, toujours symétriques. Le cryptosystème implémenté dans ces deux familles des cartes est le plus souvent le DES.

Systèmes multiprestataires utilisant la cryptographie asymétrique. L'avenir appartient à ces derniers, toutefois les cartes capables de faire de la cryptographie asymétrique, tel le RSA, sont encore trop chères.

3.4 La carte et les besoins cryptographiques.

Les cartes actuelles font facilement du cryptage DES, mais le RSA reste un problème. Le tableau suivant résume la situation. Les données sont données à titre indicatif pour une carte courante, à prix de revient avoisinant 10 F, qui ne comprend pas de coprocesseur arithmétique.

ressource	disponible	dédié à la cryptographie	cryptage DES	signature RSA
ROM	8 koctets	2 koctets	1 koctets	1 koctets
RAM	128 octets	64 octets	32 octets	192 octets
E^2PROM	4 koctets	1 koctets	8 octets	128 octets
puissance de calcul	$\sim$ 100 ms pour **une** multiplication 512 x 512 octets		prend <6 ms	demande 768 multiplications

Pour des raisons évidentes on voudrait qu'une carte à puce fasse le calcul en < 1 s. Actuellement, avec une carte sans coprocesseur fait la signature RSA en 20 à 30 s, ce qui n'est possible que dans certaines applications : p.ex. une "carte mère" qui signe des certificats pour d'autres cartes en phase de personnalisation, ou d'autres applications de haute sécurité rarement actionnées.

Dans ce contexte il y a deux possibilités :
Ou bien attendre que le progrès technologique fasse chuter le prix des composants suffisamment pour pouvoir faire du RSA sans entrave, ou bien chercher les cryptosystèmes alternatifs dans le domaine asymétrique. Cette deuxième solution nous a paru préférable.

Quatrième partie

La classe OR ("Obscure Representation").

Chapitre 4

Les cryptosystèmes de la classe OR.

L'idée de définir cette classe de cryptosystèmes, ainsi que le terme "Obscure" apparaissent au Japon vers 1983 [25], [46].

On va parler des cryptosystèmes asymétriques dans le contexte de chiffrement (clé publique, clé secrète, message clair, message chiffré etc.). Ces cryptosystèmes peuvent être utilisés en signature, et cela de façon standard.

Utilisation en signature Pour signer, on déchiffre un court $haché$ du message à signer, ainsi seul le possesseur de la clé secrète sait signer. Pour la vérification, on chiffre avec la clé publique et on compare avec le haché du message. Chacun sait vérifier la signature, grâce à la clé publique du cryptosystème.

Certain schémas sont bijectifs, dans d'autres un message chiffré admet plusieurs clairs. Si le nombre d'antécédents n'est pas trop grand, on peut utiliser ces schémas tels quels, en signature. Pour le chiffrement, on les rend injectifs, par exemple en ajoutant de la redondance au message clair.

La classe OR que nous définirons plus bas, regroupe sous un nom une large classe de cryptosystèmes asymétriques dont la difficulté repose en partie sur une instance du problème général appelé IP. Les schémas cryptographiques basent toutefois rarement leur sécurité sur IP seul.

4.1 Le problème IP (Isomorphisme des Polynômes)

Soit K un corps fini. On a $K = \mathbf{F}_{p^m}$.
Soit $\mathcal{A}$ l'ensemble de u équations quadratiques à n variables $a_1, a_2, \ldots a_n$. Posons $a_0 = 1$, ce qui permet de les écrire sous une forme compacte :

$$\mathcal{A} : \begin{cases} b_k = \sum_{i=0}^{n} \sum_{j=i}^{n} \lambda_{ijk} a_i a_j \\ \text{avec } k = 1..u \end{cases}$$

On considère deux applications linéaires inversibles
$S : K^n \to K^n$ et $T : K^u \to K^u$.
On effectue un changement de variables
$(a_1, \ldots, a_n) = S(x_1, \ldots, x_n)$ et $(y_1, \ldots, y_n) = T(b_1, \ldots, b_u)$.
Ainsi on obtient un deuxième ensemble d'équations, isomorphe au premier. On le notera $\mathcal{B}$. On a :
$\mathcal{B} = T \circ \mathcal{A} \circ S$:

$$\mathcal{B} : \begin{cases} y_k = \sum_{i=0}^{n} \sum_{j=i}^{n} \lambda_{ijk} x_i x_j \\ \text{avec } k = 1..u \end{cases}$$

Le problème IP consiste à retrouver S et T étant donné les deux ensembles d'équations $\mathcal{A}$ et $\mathcal{B}$.

4.2 Les fondements cryptologiques.

Tous les cryptosystèmes étudiés sont basés sur le paradigme suivant :

1. Le système d'équations $\mathcal{A}$ est facile à résoudre, grâce à une structure mathématique cachée.

2. Le système $\mathcal{B}$ a l'apparence d'un système quelconque de degré 2. Or, le problème de résoudre un tel système est NP-complet [14].
D'après Jacques Patarin, il existe toutefois une méthode pour résoudre ce type d'équations, appelée bases de Grobner, mais qui est praticable que jusqu'à $n = 8$.

De même, le problème de passer de $\mathcal{B}$ à $\mathcal{A}$, le problème IP, est probablement difficile, bien qu'il soit moins étudié à l'heure actuelle que le célèbre problème $P \neq NP$. Évidemment rien n'assure que récupérer S et T est nécessaire pour résoudre un tel système d'équations, et on peut craindre de pouvoir s'en passer. La sécurité dépend alors de la solidité du $\mathcal{B}$ lui même, NP-complet en général et sans aucune garantie en particulier. Un grand nombre de systèmes ont été cryptanalysés sans avoir à récupérer

S et T, et le problème de récupérer S et T en temps polynomial reste toujours ouvert, sauf pour D^*, que nous avons résolu dans 8.3.

Remarque importante : Il y a très peu de chances à ce que le problème de résoudre les équations publiques $\mathcal{B}$ soit effectivement NP-complet. En effet, un théorème connu de Gilles BRASSARD [3] dit que si résoudre un cryptosystème asymétrique utilisable efficacement en chiffrement est NP-dur alors $NP = Co - NP$. Or la plupart des schémas étudiés peuvent être rendus bijectifs et la communauté scientifique mondiale est presque convaincue que $NP \neq Co - NP$.

3. Pour les systèmes à deux étages, la sécurité peut être basé sur un troisième "pilier" : Le problème de décomposer une fonction f de degré 4 en produit de composition de deux fonctions de degré 2 : $f = g \circ h$. On sait très peu sur ce problème.
Nous verrons que la plupart des systèmes de degré 2 se révéleront faibles. Néanmoins le système HFE tient toujours, et la plupart des systèmes à 2 étages également.

4.3 Le cadre général OR.

Définition 4.3.0.1 (cryptosystème de classe OR ("Obscure Representation")) *On appelle un cryptosystème de classe OR le cryptosystème de degré 2 basé sur le problème IP précédent, contenant plusieurs formes quadratiques $\mathcal{A}_i$, qu'on appellera "les boî tes", suivant le schéma de la Figure 4.1.*
Par extension, un système OR à deux étages, ou de degré 4, est la composition en série de deux schémas de ce type.

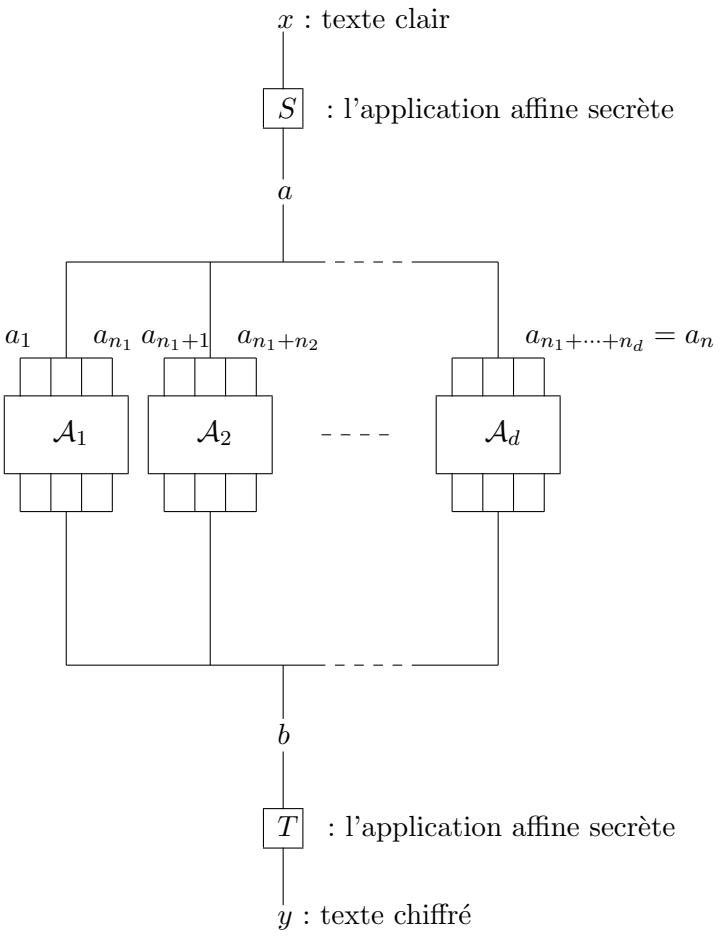

FIG. 4.1 – Schéma général OR à 1 étage (faible).

4.4 Exemples

Parmi tous les systèmes de classe OR à un seul étage connus, la plupart se sont avérés faibles. Celui qui principalement résiste à la cryptanalyse, c'est le système appelé HFE (voir 4.4.4., chapitre 9 ou [15]). Deux autres exemples c'est HM+ (chapitre. 7) et Super Scotch (chapitre 10).

4.4.1 le C^*

Le premier cryptosystème de ce type, appelé C^* fut proposé à EUROCRYPT'88 [27]. On notera K^n une extension de degré n de $K = \mathbf{F}_q$, avec $q = 2^m$. La notation provient du fait qu'une telle extension s'identifie à l'espace vectoriel de dimension n sur K en utilisant un polynôme irréductible de degré n sur K.
Soit $0 < \theta < n$ tel que $1 + q^\theta$ est premier avec $\#(K^n)^* = q^n - 1$.
L'application suivante
$K^n \to K^n$, $x \mapsto x^{1+q^\theta}$
est une forme quadratique **bijective** sur K^n. Elle est quadratique car

$$x \mapsto x^{q^\theta}$$

est linéaire, et elle est bijective, son inverse étant une puissance $\hbar$ dans K^n telle que $\hbar(1 + q^\theta) \equiv 1 \ mod \ (q^n - 1)$.

Dans C^* on suppose que n est divisé en plusieurs parts :

$$n = n_1 + \cdots + n_d$$

Le C^* consiste à prendre toutes les formes quadratiques $\mathcal{A}_i : K^{n_i} \to K^{n_i}$ de type

$$\mathcal{A}_i : \ x \mapsto x^{1+q^{\theta_i}}$$

pour différentes valeurs de θ_i tels que $1 + q^{\theta_i}$ est premier avec $q^{n_i} - 1$.

L'attaque de Jacques Patarin Le C^* avait été cassé dans [14]. L'attaque est remarquablement simple et surprenante, et consiste à remarquer que si pour la boî te $\mathcal{A}_i$ on a $b = a^{1+q^{\theta_i}}$, alors

$$ab^{q^{\theta_i}} = ba^{q^{2\theta_i}}.$$

Cette équation, dans K^{n_i}, peut se réécrire comme n_i équations dans K (on utilisera la représentation du corps pour réécrire la multiplication de K^{n_i}). Qui plus est, les équations sont bilinéaires en les a_i et les b_i. En tout en obtient $\sum n_i = n$ équations bilinéaires en les a_i et les b_i.
Maintenant, comme les a_i et les b_i sont respectivement des fonctions affines des x_i et des y_i respectivement, il existe n équations bi-affines entre les x_i et les y_i.

Celles-là, ne sont pas publiques, mais comme leur degré est peu élevé, on peut les retrouver facilement : On suppose que leur $\mathcal{O}(n^2)$ coefficients sont des variables, on utilise $\mathcal{O}(n^2)$ paires clair/chiffré (qu'on peut obtenir à partir de la clé publique) pour obtenir des équations sur ces variables, et on récupère ces coefficients par une simple réduction de Gauss.

Les équations obtenues, dont on peut montrer qu'au moins $2/3$ d'entre elles sont indépendantes ([11], version étendue de [14]), Permettent toujours en pratique de cryptanalyser des schémas réalistes. Des nombreux tests ont été effectués dans [11] et par l'auteur de ce rapport.

Ces équations permettent de récupérer le texte clair x de tout texte chiffré y : Soit un y, qu'on substitue aux équations obtenues. On a n équations linéaires à n inconnues x_i, et on peut facilement obtenir le texte clair x. Il peut arriver que les équations, après substitution, ne soient plus indépendantes, dans ce cas on retrouve le peu de bits qui restent par la recherche exhaustive, ou en générant des équations un peu plus généralisés ([11], version étendue de [14]). □

4.4.2 le D^*

Dans le D^* K est de caractéristique $p >> 2$, et les "boî tes" sont simplement des fonctions $x \mapsto x^2$. Ce système est alors "presque bijectif", car on peut extraire la racine carré au signe près. On connaî t deux attaques indépendantes sur D^*, une est exposée dans [**?**], l'autre dans le chapitre 8.
Notons cependant que pour 2 étages de D^*, qu'on appelle D^{**}, on ne sait pas cryptanalyser (2 cryptanalyses ont été publiées depuis...).

4.4.3 le HM ("Hidden Matrix")

Le HM avait été proposé, dans un cas particulier, dans [26] en 1985. Il consiste à prendre $d = 1$, n étant un carré : $n = s^2$, et la forme quadratique consiste à regarder les a_i comme les coefficients d'une matrice A de taille $s * s$ et à utiliser le carré matriciel $B = A^2$. Ceci n'est pas bijectif, mais on sait l'inverser sur un large ensemble. Le chapitre 7 explique les détails, montre comment cryptanalyser le HM et tente de le réparer, en définissant un schéma plus général - HM+.

4.4.4 le HFE ("Hidden Fields Equation")

Le HFE est un cryptosystème remarquable proposé dans [15]. Il est né d'une tentative de généraliser et "réparer" le C^*. Le HFE offre déjà par rapport au C^* l'avantage d'avoir un secret supplémentaire, car la forme quadratique est prise dans un ensemble potentiellement très large, et on ne sait pas la récupérer.
Nous décrirons la variante "Basic HFE" avec une seule équation cachée monovariable. Il repose sur l'existence des algorithmes polynômiaux pour trouver toutes les racines

d'une équation polynomiale sur un corps fini, à condition que son degré ne soit pas trop grand. Il existe plusieurs algorithmes, par exemple Berlekamp-Rabin, qui est de complexité moyenne $\mathcal{O}(mn^3d^2 \log d)$. Le HFE prévoit d'effectuer la signature sur un PC, et seulement la vérification sur des cartes à puce. Dans ce cas, on estime qu'on peut se permettre de résoudre des équations de degré allant jusqu'à 1024. La résolution pour les petits degrés (de l'ordre de 20) est envisageable même sur une carte à puce.

Comme nous avons mentionné, "les boî tes" de HFE sont des polynômes en une seule variable $x \in K^{n_i}$, où n_i est le nombre d'entrée de la boî te.
Les polynômes sont sous une forme particulière, afin de donner des équations de degré 2 :

$$f : x \mapsto \sum_{i,,j} \beta_{ij} x^{q^{\theta_{ij}}+q^{\varphi_{ij}}}$$

Ceci donne une forme quadratique, car les fonctions $x \mapsto x^{q^\sim}$ sont linéaires. Cette forme quadratique n'est évidemment plus bijective en général.

A partir du degré 17, on ne connaî t actuellement aucune d'attaque générale - voir chapitre 9.

4.4.5 Les boî tes S

Les boî tes sont des petites formes quadratiques **aléatoires** que l'on peut inverser par la recherche exhaustive. Un seul étage d'un tel schéma est faible. Par contre il devient très intéressant comme un deuxième étage d'un schéma à 2 étages [19]. On ne sait pas attaquer ce type de schéma en général.

4.4.6 Les schémas "tronqués"-Scotch

L'algorithme Scotch [15] consiste à enlever une ou plusieurs équations publiques dans C^*. C'est un schéma non-bijectif. Une attaque qui est présentée dans 10. Une variante appelée "Super-Scotch", utilisable uniquement en signature, résiste toujours à la cryptanalyse.

4.4.7 Des variantes sortantes du cadre OR.

Tous les algorithmes proposés sont susceptibles d'être composées et donner des systèmes de degré 4 intéressants, dont on sait cryptanalyser très peu [16] : les variantes où le deuxième tour est un C^*, et le premier est faible. Une autre voie c'est des systèmes de degré 3, ou/et avec $p > 2$. La plupart des schémas OR peuvent facilement être modifiées dans ce but et succombent souvent également aux attaques.
Dans [16], [17] et [18] plusieurs autres schémas conceptuellement proches de la classe OR sont étudiés, dont le "dragon", dont avons confirmé le résultat connu [11] d'existence de clés faibles et qui est cassé dans [**?**].

Chapitre 5

Propriétés de la classe OR.

Du fait d'avoir les équations publiques de degré 2, tous les cryptosystèmes de classe OR de degré 2 ont en commun un certain nombre de propriétés.

5.1 Remarques générales.

Il est intéressant de voir, que la clé publique dans la classe OR peut être récupérée à partir d'un petit nombre de couples (clair,chiffré), environ $\mathcal{O}(n^2/2)$. Dans le RSA, au contraire, lorsque l'exposant public e est grand, il est aussi difficile de récupérer la clé publique que récupérer la clé secrète (!) - c'est le problème du logarithme discret.

5.1.1 Complexité des calculs

A priori le cadre de la classe OR permet de concevoir les cryptosystèmes dont la complexité calculatoire en signature est nettement inférieure à RSA. Cela est moins net pour les systèmes de degré 2, car les principaux cryptosystèmes qui résiste aux attaques (HFE et HM+) ont des calculs qui peuvent être complexes en clé secrète. Par contre, on peut combiner 2 étages peu sûrs, et obtenir un système plus sûr, ce qui donne la clé publique de degré 4. Cela peut donner facilement des schémas avec des calculs **très** rapides qu'on ne sait pas casser à l'heure actuelle (p.ex. le double D^*).

5.1.2 La taille de la clé publique

Elle est quadratique dans le cas de systèmes de degré 2. En pratique, pour un système de signature sur 80 bits, cela donne une clé publique de l'ordre de 3 ko. Pour des systèmes à 2 étages généraux, la clé serait de 1.7 Mo, ce qui est déjà assez important, et 3 étages sortent définitivement du champ d'application.
La taille de la clé publique peut être réduite si on impose les coefficients des équations dans un sous-corps, ou si l'on prend un nombre n limité de variables. Pour un tel système à 2 étages la clé publique peut alors être ramenée à 4 ko seulement.
Tous les systèmes de la classe OR ont en commun que la clé publique est beaucoup plus

longue que la clé secrète. On peut croire que la grande redondance contenue dans la clé publique est utile au cryptanalyste, et la pratique semble montrer que oui. On peut aussi croire que le HFE est un système de degré 2 qui résiste à ce jour aux attaques connues, justement parce qu'il contre ce problème par un grand degré de liberté pour le choix de la forme quadratique.

5.2 Une attaque générique.

Cette attaque fait des hypothèses très fortes, mais qui peuvent toutefois se réaliser dans la vie réelle. Supposons qu'un même message x soit chiffré avec environ $n/2$ clé publiques différentes, ce qui donne $y^{(1)}, y^{(2)}, \dots y^{(n/2)}$. En pratique, on aurait $n/2 = 32$, et il peut effectivement arriver que le même message soit chiffré 32 fois, par exemple, si on envoie le même message à plusieurs personnes. Les équations publiques donnent alors environ $n^2/2$ équations entre les x_i et les $y_j^{(i)}$.
On considère comme nouvelles variables les x_i et les $x_i x_j$, qui sont en nombre $\mathcal{O}(n^2/2)$. Les équations, linéaires en ces nouvelles variables, sont en nombre suffisant pour les résoudre par la réduction de Gauss, et récupérer x.

Il existe une attaque du même type pour RSA, dans le cas où on utilise le même modulus n, et on chiffre le même message x avec 2 exposants publics différents e_1 et e_2 relativement premiers. On calcule alors facilement c et d tels que $1 = pgcd(e_1, e_2) = ce_1 + de_2$ et on peut récupérer x par $x = (x^{e_1})^c * (x^{e_2})^d \ mod\ n$.

Il existe aussi une autre attaque de RSA avec l'exposant 3.

On peut contrer cette attaque si on chiffre le texte clair avec une clé aléatoire DES que l'on envoie avec le message.

5.3 Falsification différentielle

Supposons qu'on connaisse la valeur de la différence de 2 messages clairs $t = x' - x$ et de leurs chiffrés correspondants $z = y' - y$. On peut alors récupérer x et x' !
Soit $y = P(x)$ l'équation publique. On écrit l'équation de la forme polaire de P :

$$Q(x,t) = P(x+t) - P(x) - P(t)$$

qui s'avère être bilinéaire symétrique en les x_i et les t_i sous la forme suivante :

$$Q(x,t) = \sum_{i,j} \beta_{ij} x_i t_j.$$

(les termes en x_i n'apparaissent pas, les termes en t_i et le terme constant non plus, car l'équation $P(x+t) - P(x) - P(t)$ s'annule quand $x = 0$, et pour $t = 0$.)
Cette équation peut être résolue par réduction de Gauss en x dès qu'on connaî t t

(et réciproquement) ! De ce fait on peut résoudre en x, avec z et t donnés, l'équation suivante :

$$z = P(x+t) - P(x) = P(t) + Q(x,t)$$

qui se ramène à

$$z - P(t) = Q(x,t).$$

Cela donne tous les x tq. la paire $(x, x') = (x, x+t)$ convient. □
Cette propriété possède également un analogue pour le RSA : Si on connaî t $y = x^e$ et $y' = (x+d)^e$ avec d connu (ce qui peut arriver par exemple quand on chiffre le même message avec une petite modification locale), alors on peut lorsque l'exposant public e n'est pas trop grand ($e < 2^{32}$) récupérer x en calculant le pgcd des polynômes $(x^e - y)$ et $((x+d)^e - y')$.

Avertissement : Il s'agit d'une attaque redoutable, à prendre toujours en considération pour la conception des cryptosystèmes à clef publique, difficile à éviter en toute généralité. Elle est particulièrement à craindre dans le contexte ou le cryptosystème est dans une carte à puce. En effet, supposons qu'on effectue deux fois la même opération de chiffrement pour le même x, et qu'on se trompe sur 1bit du texte à chiffrer à cause d'un erreur de transmission ou dans la mémoire. Une erreur de ce type apparaî tra tôt ou tard, et peut être provoquée extérieurement. L'endroit ou il y a l'erreur peut être trouvé par la recherche exhaustive, ce qui donne la différence des 2 textes clairs qu'on peut alors récupérer !
Cette attaque peut également être contrée, par l'introduction de l'aléa dans le message. Utiliser un pré-chiffrement n'est pas une très bonne solution, car l'erreur peut intervenir après.

5.3.1 Une étude plus approfondie de la résolution différentielle.

Nous avons fait de statistiques du nombre de solutions obtenues pour le résolution différentielle avec des polynômes HFE de degré fixé choisis au hasard et avec une différence z choisie au hasard. Le tableau suivant resume les résultats pour n=17, les degrés qui donnent des résultats semblables sont rassemblés.

La probabilité qu'une valeur z aléatoire donnée possède k solutions x.

k	degré 3-4	degré 5-8	degré 9-17	dg. 32-128	degré 129
0	0.667	0.760	0.751	0.749	0.750
2	0.332	0.168	0.192	0.192	0.192
4	-	0.071	0.052	0.055	0.055
8	-	-	0.002	0.0019	0.0023
16	-	-	-	0.00002	0.000013
32	-	-	-	-	$< 10^{-6}$

5.4 Attaque des multiples affines.

Il s'agit d'une généralisation de l'attaque de C^* de 4.4.1 ou [14]. Elle est depuis longtemps l'outil de base dans la cryptanalyse des système de la classe OR ([14],[15], [16], [17],[18], [11]).

Définition 5.4.0.1 (Multiple affine.) *Un multiple affine A d'une équation $f(x) - y = 0$, est une fonction $A(x, y) : K^n * K^n \rightarrow K$, affine en les x_i et de petit degré en les y_i telle que toute solution de $f(x) = y$ est aussi solution de $A(x, y) = 0$.*

L'attaque se résume à la recherche des multiples affines sur les équations publiques, et il suffit d'en trouver n environ, pour pouvoir calculer le texte clair x à partir du texte chiffré y. □

Dans notre travail nous avons donné une place particulière à 3 types de multiples affines :

type 1 Il s'agit des des équations bi-affines, c'est à dire, affines en les x_i et en les y_i (degré total 2). Pour la compacité des notations on suppose $x_0 = y_0 = 1$, et on notera par $\sim$ les coefficients (dans K), donnés explicitement qu'il est inutile de détailler ici.

$$\underbrace{\sum_{\substack{i=0..n \\ j=0..n}} \sim x_i y_j = 0}_{équations\ de\ type\ 1} .$$

type 2 Ces équations peuvent être de degré 2 en les y_i, mais leur degré total doit rester 2 :

$$\underbrace{\sum_{\substack{i=0..n \\ j=0..n}} \sim x_i y_j + \sum_{\substack{i=0..n \\ j=0..n}} \sim y_i y_j = 0}_{équations\ de\ type\ 2} .$$

type 5 Cas des équations considérablement plus longues, et plus difficiles à trouver. Pendant que le type 1 et type 2 sont de longueur quadratique, celles-là sont cubiques. C'est toujours des équations affines en les x_i, de degré 2 en les y_i, mais qui contiennent de termes mixtes de degré 3 :

$$\underbrace{\sum_{\substack{i=0..n \\ j=0..n \\ k=0..n}} \sim x_i y_j y_k = 0}_{équations\ de\ type\ 5} .$$

Remarque : Il est facile à voir que, pour le problème de recherche des multiples affines on peut supposer que le corps de base K est de type $\mathbf{F}_p$,i.e. $m = 1$. En effet, un multiple affine sur une extension de degr, α se réécrit comme α multiples affines sur le corps de base ayant les mêmes degrés en x_i et y_i.

5.4.1 Programmation de l'attaque

Dans le présent stage, un temps considérable (environ 2 mois) avait été consacré à la programmation, déboguage et optimisation des différents attaques et l'apprentissage de Borland C++ que j'ai du apprendre "sur le tas".
Le meilleur PC dont on disposait était un Pentium 100 avec 32 Mo de mémoire, ce qui permettait d'aller jusqu'à $n = 21$ dans les simulations, ce qui n'est pas gênant car nous avons observé dans la plupart des questions étudiées une transition se situant à n entre 1 et 11. La programmation était fait sous Windows, avec une gestion et allocation de la mémoire haute (au delà de 640 ko). Pour la valeur critique $n = 21$, le

programme nécessite $32Mo$ de mémoire et met environ 6 heures à tourner.
La programmation en C++ est une aventure périlleuse déconseillée au débutants et comporte des nombreux pièges très subtiles. Il faut être prêt à tout, par exemple :
#include stdio.h
main()
{char a=201, b=100 ;
unsigned int y ;
y=a%b ;
201 modulo 100
y=100*y ;
printf("%d;y)",}
Je demande aux amateurs de deviner ce que le programme va imprimer et vérifier.
C++ est un langage d'une souplesse légendaire, qui permet de créer un code très efficace. Il faut toutefois maî triser un grand nombre de détails pointilleux.

La recherche des multiples affines se ramène à chercher les dépendances linéaires entre des monômes $x_i y_j y_k$. Comme il est difficile de le faire formellement (ou vérifier pour tous les x possibles), il faut prendre un nombre suffisant de couples clair chiffré (x, y) bien distribués et sans répétitions. Pour éviter des répétitions nous avons utilisé une table de hachage simple à 2 dimensions. Pour générer les x, nous avons principalement adopté les fonctions pseudo- aléatoires standards de C++ avec petites améliorations, ce qui avait posé pas mal de problèmes. En effet certains résultats étaient trop élevés, car nous avons trouvé des équations sur les y_i seuls dus au fait que le génération n'étaient pas aléatoires.
Le recherche des multiples affines peut être un critère de sécurité pour n'importe quelle primitive cryptographique, en particulier pour les fonctions pseudo-aléatoires.
L'essentiel (95 % ou plus) du temps d'exécution des programmes se concentre sur la seule réduction de Gauss d'une énorme matrice à coefficients dans un corps fini. De ce fait, elle ne peut plus être accélérée efficacement, même si on avait une station de travail à 500 MHz. La vitesse d'exécution s'aligne à la vitesse de la mémoire - environ 20 MHz. Par contre les processeurs MMX (Multi Media Extensions) d'Intel peuvent accélérer considérablement ce type de calculs (jusqu'à 8 fois en théorie, à condition d'avoir un accès mémoire 64 bits), car ils font 8 additions de 8 bits en parallèle sur 2 registres 64 bits. Il semble en général, que pour bien de besoins cryptographiques, les Pentiums MMX pourraient surpasser les machines beaucoup plus chères. Le problème ne se pose pas pour les calculs en caractéristique $p = 2$, qui pourraient effectués 8 fois plus rapidement (environ) sur toutes les machines en utilisant tous les bits d'un octet. Nous nous sommes limités à une solution générale pour tout p.
La plupart des résultats de simulations et calculs effectuées sont exposées dans les chapitres qui suivent.

5.4.2 Généralisation de l'attaque.

On peut généraliser l'attaque des multiples affines en cherchant les multiples affines toujours affines en les x_i, mais avec des fonctions de très haut degré en les y_i. L'idée, c'est d'utiliser les formes publiques elles- mêmes et leurs itérations, avec d'autres fonctions.

Cinquième partie

Étude de 5 schémas OR.

Chapitre 6

Les systèmes à boî tes S.

L'intérêt et le défi qu'apportent des schémas à boî tes S est de proposer un schéma complètement à l'opposé de l'état de l'art actuel en cryptographie à clef publique. Dans [37], Bruce SCHNEIER soutient qu'on ne peut pas espérer de trouver des nouveaux schémas à clef publique basées sur les principes différents que ceux qu'on connaî t actuellement, et que tous les systèmes avec des polynômes devrait être regardés avec suspicion.
Aujourd'hui, seuls les systèmes basées sur des grandes structures algébriques inspirent la confiance des cryptologues.
Dans la classe OR, où on utilise une petite structure algébrique - corps fini de faible taille, et pour les boî tes S on refuse également de donner une structure algébrique à la forme quadratique. Les $\mathcal{A}_i$ sont choisies complètement au hasard.
Et pourtant on ne connaî t aucune méthode pour cryptanalyser les systèmes à boî tes S (2 étages) au delà des paramètres que nous calculerons dans ce chapitre.
Le problème principal pour les boî tes S c'est leur inversion. Elle devient possible par la recherche exhaustive, car on utilisera plusieurs petites boî tes (2,5,10 entrées).
On peut s'accommoder de la non-bijectivité des boî tes par l'ajout de la redondance, à condition que le nombre d'antécédents d'une valeur ne devienne pas trop grand.

6.1 Étude d'inversion des boî tes

Les simulations ont montré que des boî tes choisies au hasard ont des statistiques d'inversion semblables. On a ainsi calculé le nombre des y ayant $k = 0, 1, 2, \ldots$ antécédents x. Nous avons pu constater que les fonctions quadratiques se comportent exactement comme des fonctions aléatoires qui suivent la loi des probabilités $P_k = \frac{1}{ek!}$ (la démonstration de ce dernier fait est élémentaire).

Dans le tableau suivant on montre des exemples dans quelques cas intéressants ne possédants pas de multiples affines.

La probabilité qu'une valeur possède k antécédents.

k	$\frac{1}{ek!}$	4 entrées sur $\mathbf{F}_{2^4}$	5 entrées sur $\mathbf{F}_{2^3}$	3 entrées sur $\mathbf{F}_{3^3}$	5 entrées sur $\mathbf{F}_3$	2 entrées sur $\mathbf{F}_5$	1 entrée sur $\mathbf{F}_7$
0	.367879	.365341	.367431	.364680	.370370	.36	.428571
1	.367879	.368927	.367858	.370726	.370370	.36	.142857
2	.183940	.187469	.184661	.186556	.181069	.24	.428571
3	.061313	.060623	.061889	.059594	.053497	-	-
4	.015328	.014572	.014435	.015393	.020576	.04	-
5	.003066	.002487	.003051	.002591	0	-	-
6	.000511	.000518	.000610	.000406	.00411	-	-
7	.000073	.000061	.000061	.000051	-	-	-
>8	-	-	-	-	-	-	-

Ces résultats confirment la loi $\frac{1}{ek!}$ d'une fonction aléatoire.

6.2 Étude de la taille minimale des boî tes.

Le schéma étudié utilise plusieurs boî tes S. Pour la question de l'existence de multiples affines, il suffit d'étudier le schéma à une seule boî te, qu'on supposera à λ entrées, et chacune des entrés sera dans $\mathbf{F}_p^m$. On a $\lambda = n/m$.

Nous avons étudié le problème suivant :

Question : Pour quels valeurs de n, m, p, λ une boî te possède-t- elle des multiples affines, et quelle est la valeur λ minimale qui donne 0 équations ?

Dans les simulations, dont nous présenterons ici que les résultats les plus intéressants, des phénomènes de monotonicité naturels sont apparus :

1. Le nombre de multiples affines de différents types décroî t quand n croî t.

2. De même, il décroî t avec λ croissant, aussi bien à n fixé qu'à m fixé.

3. Il décroît fortement avec p. Un saut significatif accompagne déjà le passage de 2 à 3.

Tout cela permet de faire apparaî tre sur le tableau suivant que les résultats sur la frontière entre le nombre d'équation non nul et 0.

Recherche des multiples affines sur différentes tailles de S-boxes.

Function $\backslash \lambda$	1	2	3	4	5	6	7	8	9	10	11	12	13	16	18
$p = 2$															
random / $\mathbf{F}_{128}$			0 14 56												
random / $\mathbf{F}_{64}$			0 12 48												
random / $\mathbf{F}_{32}$		26 60 250	0 10 40	0 0 0											
random / $\mathbf{F}_{16}$			0 8 50	0 0 0											
random / $\mathbf{F}_8$			0 6 105	0 0 11	0 0 0										
random / $\mathbf{F}_4$					0 0 84	0 0 0									
random / $\mathbf{F}_2$						7 9 91	0 0 112	0 0 114	0 0 78	0 0 0			0 0 0		0 0 0
$p = 3$															
random / $\mathbf{F}_{19683}$		9 9 171													
random / $\mathbf{F}_{81}$		4 4 36													
random / $\mathbf{F}_{27}$		3 3 21	0 0 0												
random / $\mathbf{F}_9$	2 9 9	2 4 14	0 0 1	0 0 0											
random / $\mathbf{F}_3$	2 3 3	2 4 10	0 1 14	0 0 9	0 0 0										
$p = 5$															
random / $\mathbf{F}_{625}$	0 0 4														
random / $\mathbf{F}_{25}$	0 0 2														
random / $\mathbf{F}_5$	0 1 1	0 0 0	0 0 0												
$p \geq 7$															
random / $\mathbf{F}_7$	0 0 0														
random / $\mathbf{F}_{251}$	0 0 0	0 0 0													

LEGEND :

type 1	type 2
type 5	

Nicolas COURTOIS
Bull CP 8
juin 1997

Résultats. Ces calculs complexes donnent les résultats suivants, assez techniques, mais essentiels pour la conception de nouveaux cryptosystèmes. En effet, il est envisageable d'ajouter au moins une boî te S, dans d'autres types de schéma de classe OR, pour les renforcer.

En caractéristique 2, $\lambda \geq 4$ La boî te la plus petite qu'on ne sait pas casser possède 4 entrées sur $\mathbf{F}_{16}$, ce qui donne une boî te 16 bits $\mapsto 16$ bits.

Pour $p = 3$, $\lambda \geq 3$ suffit. La boî te la plus petite qu'on ne sait pas casser possède 3 entrées sur $\mathbf{F}_{27}$, ce qui donne une boî te 15 bits $\mapsto 15$ bits.

Pour $p = 5$, $\lambda \geq 2$ La boî te la plus petite qu'on ne sait pas casser possède 2 entrées sur $\mathbf{F}_5$, ce qui donne une boî te 6 bits $\mapsto 6$ bits.

Pour $p \geq 7$, $\lambda \geq 1$ La boî te la plus petite qu'on ne sait pas casser par les multiples affines possède 1 seule entrée sur $\mathbf{F}_7$, ce qui donne une boî te 3 bits $\mapsto 3$ bits.

Remarque. On constate que la taille effective des boî tes diminue avec p, ce qui amène à la conclusion que la sécurité augmente avec la taille de la plus petite structure algébrique sous-jacente F_p. Cela va dans le sens de l'expérience cryptographique actuelle, par exemple le RSA utilise une structure particulièrement grande $\mathcal{Z}_n$ avec n très grand.

Nouveaux schémas. A partir des résultats sur S boî tes, des nouveaux schémas asymétriques très prometteurs sont proposés dans [17]. Nous ne les avons pas cryptanalysés et nous ne les détaillerons pas ici.

Le fait que des boî tes S quadratiques ait un comportement statistique (6.1.) d'une fonction choisie au hasard est un bon argument qui suggère qu'il est possible de concevoir ainsi les schémas réellement solides.
On peut envisager des tests statistiques supplémentaires.

Chapitre 7

Le schémas HM et HM+

7.1 Le HM et son attaque.

L'idée de HM est de MATSUMOTO [26]. Rappelons que c'est un schéma où n est un carré $n = s^2$, les éléments de K^n sont vus comme des coefficients d'une matrice sxs, et la forme quadratique est simplement la multiplication matricielle :

$$B = A^2.$$

À priori c'est un schéma intéressant par sa simplicité, et la facilité extrême de calculs, aussi bien en clé secrète qu'en clé publique. Le principal inconvénient réside toutefois dans une non-bijectivité, mais cela n'est pas grave. Premièrement, un schéma non-bijectif convient parfaitement en signature, deuxièmement on peut espérer le rendre bijectif. Par exemple pour $n = 2$, on utilise le fait que A satisfait sa propre équation caractéristique, qui s'écrit :

$$A^2 - tr(A)A + dét(A) = 0.$$

Il suffit alors que $tr(A) \neq 0$ pour inverser la fonction. En plus si $tr(A) \neq 0$ alors $tr(B) \neq 0$ et donc on obtient une bijection de l'ensemble

$$\{A|tr(A) = 0\}$$

dans lui-même.
L'équation caractéristique, compte tenu de la présence du déterminant est de degré élevé $(\sqrt{n})$. Elle n'est donc pas d'un grande utilité pour obtenir ou assurer l'existence des multiples affines.

Malgré cela, le HM n'est pas sûr.

Il nous a paru assez étonnant de trouver que l'équation suivante :

$$AB = BA$$

suffit pour cryptanalyser le système (!) En effet, le produit matriciel n'est pas commutatif et cette équation nous donne n multiples affines. La dimension exacte de l'espace d'équations obtenues est assez complexe à décrire [8], mais elle est proche de n, et par conséquent on peut retrouver compléter par la recherche exhaustive. évidemment, un nombre proche de n d'équations indépendantes ne doit pas forcément en donner autant une fois y substitué dans ces équations. En effet, l'équation $AB = BA$ peut aussi servir à résoudre $B = A^3$ et d'autres équations polynomiales. On peut montrer que les équations qui satisfont $AB = BA$ sont exactement les équations polynomiales, ce qui permet de prédire qu'il faudra faire la recherche exhaustive sur environ $\sqrt{n}$ x_i supplémentaires, car la dimension de l'espace de tous les polynômes en A est $\sqrt{n}$ (grâce à l'équation caractéristique qui est de degré $\sqrt{n}$).
La recherche exhaustive n'est pourtant pas nécessaire car l'expérience a montré, voir les résultats à la page suivante, qu'il existe **beaucoup** plus que n de multiples affines de type 2 et de type 5 dont le nombre nous n'avons pas su justifier. On s'aperçoit aussi, que pour $p = 2$ on peut avoir jusqu'à $(n+1)$ multiples affines de type 1. Cette équation supplémentaire s'explique comme suit :

$tr(B) = tr(A^2) = tr(A)^2 = tr(A)$, la dernière équation provient du fait qu'on soit dans $\mathbf{F}_2$).

7.2 le HM+ ou comment réparer le HM

L'idée est de prendre l'équation $B = A^2 + MA$ où M est une matrice quelconque. On obtient alors effectivement 0 multiples affines à partir de $p = 7$, ce qui montre le tableau suivant :

Recherche des multiples affines sur HM et HM+.

$p\backslash n$	$2x2$	3x3	4x4
	HM, $B = A^2$		
p=2	10 16 39	10 18 153	17 32 292
p=3	4 4 28	9 9 89	16 16 271
p=31	3 3 14	8 8 79	15 15 254
p=127	3 3 14	8 8 79	15 15 254
	HM+, $B = A^2 + MA$ M-random matrix		
p=2	10 16 39	3 11 133	3 18 49
p=3	1 1 14	1 1 11	1 1 18
p=5	0 0 1	0 0 1	0 0 1
p=7	0 0 0	0 0 0	0 0 0

LEGEND : type 1 type 2 / type 5

7.2.1 Problème de l'inversion de HM et HM+

L'équation $B = A^2$ se résout facilement grâce à l'équation caractéristique où l'on peut mettre A en facteur dans des termes où la puissance de A est impaire, et ainsi on peut se ramener à la division de deux matrices, fonctions de B.

L'équation $B = A^2 + MA$ de HM+ peut également être résolue en temps polynomial [31],la méthode est assez longue à décrire.

Nous avons effectué des simulations concernant le degré d'inversion de ces fonctions. Le tableau suivant montre des résultats pour des matrices 3x3. Il va sans dire que pour le HM+, la matrice M avait été choisie au hasard et nous nous sommes assurés que le comportement est stable et diffère peu du comportement typique présenté :

Nombre d'antécédents	HM	HM+	HM	HM+	HM	HM+
	3x3 sur $\mathbf{F}_2$		3x3 sur $\mathbf{F}_3$		3x3 sur $\mathbf{F}_5$	
0 x	252	280	14313	10841	1388180	1129810
1 x	160	120	0	3768	0	240721
2 x	42	56	3990	3053	368960	394005
3 x	0	24	0	815	0	53060
4 x	56	12	702	495	184450	67193
5-10 x	0	18	221	629	0	59924
11-30 x	2	2	455	79	2294	8057
31-150 x	-	-	1	3	9238	352
151-250 x	-	-	1	-	0	3
251-2000 x	-	-	-	-	3	-
$>$2000 x	-	-	-	-	-	-

7.2.2 Conclusion.

Le HM et HM+ sont comparables au niveau de degré d'inversion. Il y a toutefois moins de chances de rendre HM+ bijectif, car cela risque de donner de l'information sur la matrice secrète M. Ils sont relativement "peu" bijectifs (moins que le HFE par exemple), ce qui oblige à faire plus de calculs en déchiffrement ou en signature et rend l'algorithme moins efficace. Nous ne connaissons à ce jour aucune attaque qui fonctionne pour le HM+ pour $p \geq 7$. De plus, le schéma donne des signatures **très** faciles à vérifier. Même la carte à puce la moins chère au monde saurait faire ces calculs, il s'agit en fait des équations publiques quadratiques, qui nécessitent $\mathcal{O}(n^3/2)$ multiplications très simples, dans $\mathbf{F}_p$. Dans un exemple concret, $n = 80$, le temps de calcul est estimé à 0.3 s avec 3 cycles d'horloge à $3,57MHz$ par multiplication. On peut également envisager de signer avec une carte, dans le cas où plusieurs petites matrices sont utilisées.

Chapitre 8

Étude et attaque de D*.

Nous l'expliquerons sur un exemple. Tous les calculs sont réalisés en MAPLE.

8.1 Un exemple de D^*.

Le D^* utilise un corps fini $\mathbf{F}_{p^n}$, $p \neq 2$ tq. $p^n \equiv 3\ mod\ 4$. Dans l'exemple $p = 3$, $n = 3$. Ce corps sera représenté comme un espace vectoriel sur $\mathbf{F}_3$, à l'aide d'un polynôme irréductible. On prendra par exemple $\mathcal{Z}^3 + 2\mathcal{Z} + 1$.
On utilisera les notations (c_1, c_2, c_3) qui équivaut à $c_1 + c_2\mathcal{Z} + c_3\mathcal{Z}^2$ dans le quotient.
Dans le cas $p^n \equiv 3\ mod\ 4$, la racine carrée se calcule facilement :

$$\sqrt{b} = b^{\frac{p^n+1}{4}} = b^7 = b^4 * b^2 * b$$

(ce qui rend les opérations secrètes très commodes pour être utilisées dans une carte à puce).
La transformation publique est une fonction $x \in \mathbf{F}_{p^n} \mapsto y \in \mathbf{F}_{p^n}$ dont la véritable structure est cachée :

$$y = T(b), b = a^2, a = S(x).$$

S et T sont des applications affines inversibles $\mathbf{F}_p^n \mapsto \mathbf{F}_p^n$. Par exemple :

$$a = \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 1 \\ 2 & 2 & 2 \\ 1 & 0 & 2 \end{bmatrix} (x_1, x_2, x_3)$$

$$y = \begin{bmatrix} 2 \\ 0 \\ 2 \end{bmatrix} + \begin{bmatrix} 2 & 0 & 1 \\ 2 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} (b_1, b_2, b_3)$$

La fonction interne $b = a^2$ peut être écrite comme une forme quadratique $\mathbf{F}_p^n \mapsto \mathbf{F}_p^n$, facile à calculer explicitement :

$b_3\mathcal{Z}^2 + b_2\mathcal{Z} + b_1 =$

$\left(a_3\mathcal{Z}^2 + a_2\mathcal{Z} + a_1\right)^2 \ mod\ \left(\mathcal{Z}^3 + 2\mathcal{Z} + 1\right) =$

${a_3}^2\mathcal{Z}^4 + 2\,a_3\mathcal{Z}^3a_2 + 2\,a_3\mathcal{Z}^2a_1 + {a_2}^2\mathcal{Z}^2 + 2\,a_2\mathcal{Z}a_1 + {a_1}^2 \ mod\ \left(\mathcal{Z}^3 + 2\mathcal{Z} + 1\right) =$

$(2\,a_3a_1 + {a_2}^2 + {a_3}^2)\mathcal{Z}^2 + (2\,a_2a_1 + 2\,{a_3}^2 + 2\,a_3a_2)\mathcal{Z} + {a_1}^2 + a_3a_2.$

ce qui donne :

$$\mathcal{A} : \begin{cases} b_1 = {a_1}^2 + a_2a_3 \\ b_2 = 2\,a_2a_1 + 2\,{a_3}^2 + 2a_2a_3 \\ b_3 = 2\,a_3a_1 + {a_2}^2 + {a_3}^2 \end{cases}$$

Maintenant on effectue le changement de variables et on obtient une autre forme quadratique $\mathbf{F}_p^n \mapsto \mathbf{F}_p^n$, qui constitue l'équation publique :

$$\mathcal{B} : \begin{cases} y_1 = x_3x_2 + {x_2}^2 + x_1x_3 + x_1 + x_3 + {x_3}^2 + {x_1}^2 + 2\,x_2 \\ y_2 = 2\,x_1x_2 + x_3x_2 + {x_2}^2 + 2\,x_1x_3 + 2\,x_3 + 2\,{x_1}^2 + 2\,x_2 \\ y_3 = 1 + {x_2}^2 + 2\,x_3 + {x_3}^2 + 2\,{x_1}^2 + x_2 \end{cases}$$

Dans ce qui suit on n'utilisera **que** l'information contenue dans les coefficients de cette équation. Notamment, on va montrer comment on peut inverser cette fonction quadratique sans connaî tre S et T.

8.2 Une attaque.

Le premier pas de l'attaque consiste à écrire des applications affines S et T comme des équations sur $\mathbf{F}_{p^n}$. Toute application affine $\mathbf{F}_p^n \to \mathbf{F}_p^n$:

$$x \mapsto constante + M * (x_1, \ldots, x_n),$$

où M est une matrice dans $\mathcal{M}_{n*n}(\mathbf{F}_p)$, peut s'écrire dans $\mathbf{F}_{p^n}$ comme suit :

$$a = M_0 + \sum_{i=1}^{n} x_iM_i.$$

M_0 dénote le terme constant, et M_i est la i-ème colonne de la matrice M vue comme un vecteur de $\mathbf{F}_p^n$ et donc comme un élément de $\mathbf{F}_{p^n}$.

Le cryptosystème D^* s'écrit alors en une seule équation :

$$\left(S_0 + \sum_{i=1}^{n} x_iS_i\right)^2 = T_0 + \sum_{i=1}^{n} y_iT_i.$$

Notons que puisque les applications affines S et T sont inversibles, les S_i et les T_i sont linéairement indépendants et forment deux bases de $\mathbf{F}_{p^n}$.

On va noter par $\sim$ les coefficients qu'on calcule explicitement à partir des valeurs données.
Dans l'équation précédente on substitue aux y_i des formes publiques, fonctions quadratiques des x_i, et on obtient une équation de degré 2 en les x_i, vraie pour tout x (!) :

$$S_0^2 + \sum_{i=1}^{n} 2S_0 S_i x_i + \sum_{i=1}^{n} S_i^2 x_i^2 + \sum_{i \neq j} 2S_i S_j x_i x_j =$$

$$= T_0 + (\sum \sim T_i) + \sum_{i=1}^{n} (\sum \sim T_i) x_i + \sum_{i=1}^{n} (\sum \sim T_i) x_i^2 + \sum_{i \neq j} (\sum \sim T_i) x_i x_j$$

Où à chaque fois $(\sum \sim T_i)$ est une certaine combinaison linéaire **connue et explicitement écrite** des $T_i, i = 1, 2, \ldots, n$, sans T_0, à coefficients dans $\mathbf{F}_p$.

Ce qui, en comparant les coefficients de chacun des termes en $1, x_i$ et $x_i x_j$ va donner $(n+1)(n+2)/2$ équations dans $\mathbf{F}_{p^n}$:

$$\begin{cases} S_0 = T_0 + \sum \sim T_i \\ \vdots \\ 2S_0 S_i = \sum \sim T_i \\ \vdots \\ S_i^2 = \sum \sim T_i \\ \vdots \\ 2S_i S_j = \sum \sim T_i \end{cases}$$

Remarquons que, quitte à modifier des S_i et des T_i, on peut supposer $S_0 = 1$.

Ces équations représentent la **totalité** de l'information contenue dans les formes publiques de D^*, et il y a $(n+1) * (n+2)/2$ équations avec $(2n+2)$ inconnues, ce qui fait beaucoup plus d'équations que d'inconnues et reflète le fait que la clé publique est beaucoup plus longue que la clé secrète. On peut craindre qu'il existe une méthode pour les résoudre, car avec seulement n équations supplémentaires il devient facile de le résoudre par une réduction de Gauss sur des variables T_i et $Z_{i,j} = S_i S_j$, en supposant $S_0 = 1$.
On verra plus loin comment les résoudre.
Cela n'est pourtant pas nécessaire pour la cryptanalyse de D^*.

Dans l'exemple concret pris au début du chapitre on obtient les équations suivantes :

$$\begin{cases} 1 = T_0 + T_3 \\ S_1 = 2T_1 \\ S_2 = T_1 + T_2 + 2T_3 \\ S_3 = 2T_1 + T_2 + T_3 \\ S_1^2 = T_1 + 2T_2 + 2T_3 \\ S_1 S_2 = T_2 \\ S_1 S_3 = 2T_1 + T_2 \\ S_2^2 = T_1 + T_2 + T_3 \\ S_2 S_3 = 2T1 + 2T_2 \\ S_3^2 = T_1 + T_3 \end{cases}$$

Maintenant, nous avons la possibilité d'obtenir les T_i en fonction des S_i, en utilisant n de ces équations. Elles sont certainement inversibles, car les S_i et les T_i forment deux bases de $\mathbf{F}_{p^n}$.

Dans l'exemple on obtient :

$$\begin{cases} T_1 = 2\,S_1 \\ T_2 = 2\,S_2 + 2\,S_3 \\ T_3 = 2\,S_3 + 2\,S_1 + S_2 \end{cases}$$

Ensuite, on élimine les T_i dans les $n(n+1)/2$ équations suivantes :

$$\begin{cases} {S_1}^2 = 2S_3 \\ S_1 S_2 = 2\,S_2 + 2\,S_3 \\ S_1 S_3 = S_1 + 2\,S_2 + 2\,S_3 \\ {S_2}^2 = S_1 + S_3 \\ S_2 S_3 = S_1 + S_2 + S_3 \\ {S_3}^2 = S_1 + 2\,S_3 + S_2 \end{cases}$$

Ainsi on obtient une table de multiplication complète dans la base $(S_i)_{i=1,2,\ldots,n}$. La complexité de cette partie de prétraitement de l'attaque est celle de la réduction de Gauss pour obtenir des T_i,i.e. $\mathcal{O}(n^3)$.

Grâce à cette table on peut évaluer n'importe quel polynôme sur des valeurs symboliques, on peut aussi comparer deux expressions. Cela donne une large classe d'algorithmes.
En particulier on peut inverser symboliquement la fonction cachée du cryptosystème $x \mapsto x^2$ et déchiffrer (!).

8.2.1 Exemple de déchiffrement.

On va déchiffrer le message

$$y = (2, 1, 2).$$

On écrit

$$b = T_0 + 2T_1 + T_2 + 2T_3$$

En utilisant la seule équation contenant T_0 et l'expression des T_i en fonction des S_i on obtient :

$$b = 1 + 2T_1 + T_2 + T_3 = 1 + S_3$$

Maintenant on calcule (formellement) dans la base S_i :

$$a = \sqrt{b} = b^7 = (1 + S_3)^7 = ...$$

On utilise la méthode classique 'square and multiply' :

$$b^2 = (1 + S_3)^2 = 1 + S_3 + S_1 + S_2$$

$$b^4 = (1 + S_3 + S_1 + S_2)^2 = 2\,S_3 + 2\,S_1 + 1 + S_2$$

$$b^6 = (2\,S_3 + 2\,S_1 + 1 + S_2)\,(1 + S_3 + S_1 + S_2) = (1 + S_1 + 2\,S_3)$$

$$a = b^7 = (1 + S_1 + 2\,S_3)\,(1 + S_3) = (1 + S_1 + S_2)$$

Ce qui donne le message clair $x = (1, 1, 0)$. On vérifie facilement que l'équation publique donne bien la valeur $y = (2, 1, 2)$ au point $x = (1, 1, 0)$.

La longueur des expressions ne croî t pas trop car on se ramène toujours à une forme normale $1 + \sum_{i=1}^{n} S_i$. Ces calculs ne différent en rien des calculs normalement effectuées dans un corps fini, sauf que la représentation est un peu plus lourde à utiliser.
L'attaque décrite marche sans modifications majeures pour une variante de D^* avec un polynôme de degré 2 quelconque au lieu de $x \mapsto x^2$. Il est facile à voir que modulo un changement de S et T, on peut supposer sans perte de généralité que la fonction interne est $x \mapsto x^2$.

Une autre attaque contre ce schéma avait déjà été proposée dans [19]. Toutefois, le problème de retrouver les applications affines S et T n'a jamais été résolu en temps polynomial pour aucun schéma de degré 2. Nous allons voir comment le faire dans le cas de D^*.

8.3 Comment trouver les applications affines S et T dans D^*.

On va continuer sur l'exemple précédent pour illustrer les calculs à faire.

L'algorithme comporte une partie probabiliste. Prenons n'importe quel $x \in \mathbf{F}_{p^n}$, par exemple $x = (0, 0, 1)$. On s'intéresse à l'ordre de $a = S(x)$. La plupart des éléments d'un corps fini, son groupe multiplicatif étant cyclique, sont générateurs (il y a en fait $\varphi(p^n - 1)$ générateurs). Le nombre d'essais, qu'on aurait à faire sur x, serait très petit, (de l'ordre de 2).
Pour le x choisi on a

$$a = S(x) = 1 + S_3.$$

Nous allons calculer la suite $(a, a^2, \ldots, a^{n+1})$, supposée donc être apériodique, en prenant soin de tout exprimer sous la forme $1 + \sum \sim S_i$, comme dans la partie attaque de D^*.

$$(a, a^2, a^3, a^4) = (1 + S_3, 1 + S_1 + S_2 + S_3, 1 + S_1 + 2S_2 + S_3, 1 + 2S_1 + S_2 + 2S_3).$$

Maintenant, on ne regarde que la partie non constante $\sum \sim S_i$ de ces expressions. On a $(n+1)$ termes écrits **explicitement** dans une base à n éléments, et par conséquent on peut détecter en $\mathcal{O}(n^3)$ une dépendance linéaire. Dans l'exemple on trouve facilement que

$$(S_1 + 2S_2 + S_3) + (2S_1 + S_2 + 2S_3) = 0.$$

Quand on tient compte, en plus, des termes constants, on obtient une équation polynomiale de degré au plus $(n + 1)$ en a. Ici c'est :

$$a^3 + a^4 = 2.$$

Une telle équation peut être résolue dans $\mathbf{F}_p^n$.
De nombreux algorithmes existent [8] et permettent d'obtenir **toutes** les racines de ce polynôme. Leur complexité est polynomiale tant que le degré de polynôme reste lui-même polynomial, ce qui est le cas ici. Ainsi, par exemple l'algorithme classique de Berlekamp-Rabin permet de le faire en moyenne en $\mathcal{O}(n^3 \log n \log p)$ opérations.
On obtient au plus $(n + 1)$ racines. Dans le cas présent, la fonction Roots de MAPLE donne 4 solutions distinctes :

$$a \in \{1, (1, 1, 2), (2, 0, 2), (1, 2, 2)\}.$$

Faut-il encore savoir trouver la bonne.

Chacun des $(n+1)$ choix possibles donne n ou plus équations sur les $S_1, \ldots, S_n$, car on connaî t l'expression de

$$a^m = 1 + \sum \sim S_i,$$

pour $k = 1..(n+1)$ et on saurait l'obtenir pour d'autres a^k si nécessaire.

À partir d'une valeur candidate $a^{(i)}$, on calcule donc (dans $\mathbf{F}_{p^n}$) ses n puissances successives, dont l'expression en $1+\sum \sim S_i$ a déjà été calculée, et on obtient n équations linéaires en les S_i, qu'on résout dans $\mathbf{F}_{p^n}$ par la réduction de Gauss en $\mathcal{O}(n^3)$. Sur l'exemple, pour la troisième valeur $a^{(3)}$, on obtient le système de n équations suivant :

$$(*) \begin{cases} 1 + S_3 &=& \left(a^{(3)}\right)^1 &=& (2,0,2) \\ 1 + S_1 + S_2 + S_3 &=& \left(a^{(3)}\right)^2 &=& (1,2,0) \\ 1 + S_1 + 2S_2 + S_3 &=& \left(a^{(3)}\right)^3 &=& (1,2,2) \end{cases}$$

Comme on a supposé que a était générateur, il est facile à montrer que le système aura une unique solution.
En effet, soit k, le plus petit entier tel que les $\{(a-1), (a^2-1), \ldots, (a^k-1)\}$ soient linéairement dépendants.

D'une part $k \leq (n+1)$. D'autre part, on a

$$(a^{k+1}-1) = a(a^k-1) + (a-1) = a(\sum_{j=1}^{k-1} \sim (a^j-1)) + (a-1) =$$

$$= (\sum_{j=1}^{k-1} \sim (a^{j+1} - 1 - a + 1)) + (a-1)$$

ce qui se réécrit encore sous la forme $\sum_{j=1}^{k} \sim (a^j-1)$, et donc aussi comme $\sum_{j=1}^{k-1} \sim (a^j-1)$.
Il est facile à voir que pour tout m on peut toujours écrire (a^m-1) comme $\sum_{j=1}^{k-1} \sim (a^j-1)$. Et donc, puisque a est générateur, tout élément de $\mathbf{F}_{p^n}$, sauf peut-être -1, est dans l'espace vectoriel engendré par $(a-1), (a^2-1), \ldots, (a^{k-1}-1)$ qui doit être de dimension n au moins.

Donc $k \geq (n+1)$ et $k \leq (n+1$. Finalement $k = (n+1)$ et les (a^i-1), $i = 1..n$ forment une base de $\mathbf{F}_{p^n}$, ce qui assure l'unicité de la solution du système d'équations $(*)$. Pour le résoudre on fait une réduction de Gauss sur des équations $(a^i-1) = \sum \sim S_j$. Dans l'exemple cela donne :

$$S_1 = (2, 2, 2)$$
$$S_2 = (0, 0, 2)$$
$$S_3 = (1, 0, 2)$$

Une fois S connu, il est facile de calculer T.
On peut donc vérifier laquelle des solutions est bonne. En pratique, il est inutile, de répéter cette résolution pour toutes les valeurs candidates $a^{(i)}$, ce qui donnerait en tout $\mathcal{O}(n^4)$ opérations. Il vaut mieux générer un autre polynôme en a, avec quelques autres puissances de a, et éliminer ainsi des a possibles. On calcule le pgcd de deux polynômes, et vraisemblablement on n'aura pas a appliquer les méthodes de résolution des équations polynomiales, car très vite on aura tombé à degré 1 ou 2. On est certain d'avoir les bonnes valeurs de S_i dès qu'on trouve un seul a comme solution.

Dans la solution générale il ne faut pas oublier qu'on a supposé $S_0 = 1$. En toute généralité, la solution générale est

$$(\gamma S_0, \gamma S_1, \ldots, \gamma S_n) \quad et \quad \left(\gamma^2 T_0, \gamma^2 T_1, \ldots, \gamma^2 T_n\right), \ pour \ \gamma \in \mathbf{F}_{p^n}.$$

Vérifions que les S_i obtenus sont, à une multiplication près, les mêmes qu'on avait pris au début. Soit $\gamma = (1, 0, 2)$:

$$\begin{aligned}
&(\gamma * 1, \gamma * S_1, \gamma * S_2, \gamma * S_3) = \\
&\quad ((1,0,2), (1,0,2) * (2,2,2), (1,0,2) * (0,0,2), (1,0,2) * (1,0,2)) = \\
&\quad ((1,0,2), (1,2,1), (0,2,0), (1,2,2))
\end{aligned}$$

□

Chapitre 9

HFE et tentatives d'attaque.

Nous n'avons pas réussi, malgré des efforts intenses à casser le HFE. Il semble que sa difficulté est liée au nombre de monômes de l'équation polynomiale secrète.

Nous avons confirmé les résultats de [11] sur l'existence des multiples affines pour certains cas de HFE, ce qui se lit sur le tableau ci-joint. Il semble que cette faiblesse soit liée au nombre et aux degrés de monômes du polynôme utilisé. On a pu toutefois casser certains assez longs polynômes, p.ex. $x^{18} + x^{32} + x^{64} + x^{66} + x^{80} + x^{128}$, $x + x^2 + x^3 + x^5 + x^{16} + x^{17} + x^{18} + x^{20}$ ou $x + x^3 + x^5 + x^6 + x^{10} + x^{16} + x^{20} + x^{24}$.

Nous avons essayé d'appliquer la méthode du chapitre 8 pour le C^* et le HFE. Sans succès, il reste que cette voie peut aider la cryptanalyse à cause de l'utilisation 'entière' de l'information contenue dans les équations publiques.

Si le polynôme interne de HFE est publique, il suffirait de savoir résoudre le IP. Le chapitre suivant montre des nouvelles méthodes pour ce faire, qui montrent que IP est beaucoup moins difficile qu'on ne le croyait. Notre attaque en $\mathcal{O}(q^{n/2})$ montre que dans le cas où le polynôme est publique, les applications S et T linéaires, un HFE 80 bits n'est pas solide.

Tant que le polynôme interne est secret, le HFE pourrait s'avérer solide même si on sait résoudre le problème IP avec autant de succès que le fameux problème d'isomorphisme de graphes (lié à IP, voir [19]).

Le HFE reste un cryptosystème très intéressant, car il a l'avantage de donner des signatures très courtes de l'ordre de 128 bits, et même de 64 bits dans certaines variantes. Le meilleur schéma actuellement utilisé est celui de Schnorr, avec des signatures sur 220 bits [44].

Multiples affines dans HFE comparé à d'autres fonctions.

Function $\backslash n$	9	11	12	13	14	16	18	20	24	30	48	64	80	128
random S-box														
$\lambda = 2$ / $\mathbf{F}_{1024}$								75 220						
$\lambda = 3, 4$ / $\mathbf{F}_{16}$			0 8 50			0 0 0								
$\lambda = n$ / $\mathbf{F}_2$	0 0 78	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0				
$C^*, p = 2$														
$x \mapsto x^3$		22 22 264 b	24 24 312 (3)	26 26 364 b	28 28 420 (3)	32 32 544 (3)	36 36 684 (3)	40 40 840		60	96	128	160	256
HFE, $p = 2$														
$x^2 + x^3 + x^{16}$		0 1 66 (5)	0 3 18 (7)	0 1 1 (5)	0 1 1 (10)	0 1 1 (6)	0 3 3 (16)	0 1 1	0 3 3	0 3 3		0		0
$x^3 + x^5 + x^{12}$		1 1 12 (4)	1 1 13 (11)	1 1 14 (5)	1 1 15 (6)	1 1 17 (7)	1 1 19 (10)	1 1 21	1 1 25	1 1 31			1	1
$x + x^3 + x^5$	0 0 79	0 0 66 b	0 0 72 (5)	0 0 78 b	0 0 84 (5)	0 0 96 (5)	0 0 108 (5)	0 0 120	0 0 144	0 0 180			0	0
$x + x^5 + x^6 + x^9 + x^{12}$	0 0 78	0 0 0 (12)	0 0 0 (10)	0 0 0 (6)	0 0 0 (7)	0 0 0 (8)	0 0 0	0 0 0	0 0 0	0 0 0			0	0
$x + x^3 + x^5 + x^6 + x^9$	0 0 78	0 0 0 (6)	0 0 0 (9)	0 0 0 (5)	0 0 0 (7)	0 0 0 (9)	0 0 0	0 0 0	0 0 0	0 0 0			0	0
$x^6 + x^{10} + x^{24}$	1 1 81	1 1 12 (4)	1 1 13 (11)	1 1 14 (5)	1 1 15 (6)	1 1 17 (7)	1 1 19	1 1 21	1 1 25	1 1 31			1	1
$x^5 + x^{10} + x^{24}$	0 0 78	0 0 0 (6)	0 0 0 (19)	0 0 0 (5)	0 0 0 (6)	0 0 0 (7)	0 0 0	0 0 0	0 0 0	0 0 0			0	0
$x^{10} + x^{12} + x^{20}$		1 1 45 (5)	5 5 65 (10)	1 1 53 (5)	1 1 57 (7)	1 1 65 (8)		1 1 81	5 5 125	5 5 155			1	1
$x + x^5 + x^9$		0 0 0 (4)	0 0 0 (5)	0 0 0 (4)	0 0 0 (6)	0 0 0 (6)		0 0 0		0 0 0			0	0
$x^6 + x^{17}$		0 0 0 (5)	0 0 0 (6)	0 0 1 (6)	0 0 0 (7)	0 0 0 (7)		0 0 0	0 0 0	0 0 0			0	0
$x^1 + x^2 + x^3$		22 22 264 b	24 24 312 (3)	26 26 364 b	28 28 420 (3)	32 32 544 (3)		40 40 840	48 48 1200	60 60 1860			160	256
$x^{18} + x^{32} + x^{64} + x^{66} + x^{80} + x^{128}$		1 1 99 (6)	3 3 39 (10)	1 1 14 (10)	1 1 15 (8)	1 1 17 (10)		1 1 21	1 1 25	1 1 31			1	
$x^3 + x^5 + x^8$		1 1 45 (6)	2 2 50 (7)	1 1 53 (6)	2 2 58 (6)	2 2 66 (8)		2 2 82	2 2 98	2 2 122			2	

LEGEND :

type 1	type 2
type 5	
(id)	

Nicolas COURTOIS
MS-LabInf
mars 1998

Chapitre 10

La cryptanalyse du Scotch.

Scotch, comme nous l'avons déjà mentionné, est un C^*, où on enlève un petit nombre $r < 10$ d'équations publiques, utilisé en signature.

L'attaque que nous présenterons est fruit d'une réflexion commune avec Jacques Patarin et Louis Goubin.

Un article entier est consacré au problème d'enlever des équations à C^* ou HFE [20].
L'attaque est exponentielle en r, on supposera que 2^r est petit.

10.1 Principe de l'attaque.

On notera P la forme publique entière de C^*, et $P_{(r+1)..n}$ la partie dont on dispose. Le but est de récupérer les équations publiques $P_{1..r}$, car dans ce cas l'attaque classique (4.4.1) de C^* s'applique. Il va de soi que on ne peut les récupérer que modulo une transformation linéaire,i.e. on va chercher d'autres équations qui engendrent le même espace d'équations.

On utilise la forme polaire de P, définie déjà dans 5.3. ;

$$Q(x,t) = P(x+t) - P(x) - P(t).$$

Sachant que P provient d'un C^*, la forme Q permet de récupérer P. Nous savons que'au coefficient :

$\alpha x_i x_j$	dans P correspond
$\alpha x_i t_j + \alpha t_i x_j$	dans Q.

Ainsi on peut récupérer tous les coefficients $\alpha x_i x_j$ de P. Ainsi on peut écrire

$$y_k = P_k(x_1, \ldots, x_n) = \tilde{P}_k(x_1, \ldots, x_n) + \sum_i \nu_{ik} x_i \qquad (1 \leq k \leq r)$$

où il reste seulement des coefficients ν_{ik} à déterminer. On posera $P_k = \tilde{P}_k$ pour $k \geq r+1$ et on notera $\tilde{y}_k = \tilde{P}_k(x_1, \ldots, x_n)$ pour tout k.

Par la cryptanalyse de C^* (4.4.1) il existe des équations de la forme :

$$\sum_{ij} \lambda_{ij} x_i y_j + \sum_i \alpha_i x_i \sum_i \beta_i y_i + \delta_0 = 0$$

et donc après y avoir remplacé des y_i :

$$\sum_{ij} \lambda_{ij} x_i \tilde{y}_j + \sum_i \alpha_i x_i \sum_i \beta_i \tilde{y}_i + \delta_0 + \sum_i \sum_{j=1}^{r} \lambda_{ij} x_i (\sum_k \nu_{kj} x_k) + \sum_{i=1}^{r} \beta_i (\sum_k \nu_{ki} x_k) = 0$$

Dans ces équations on remplace les $\tilde{y}_i$ par leurs expressions connues. On obtient une équation vraie pour tout x. Si on regarde des termes en $x_i x_j x_k$ avec i, j, k deux à deux différents, on obtient $\frac{n(n-1)(n-2)}{6}$ équations linéaires sur les n^2 inconnues λ_{ij}, ce qui suffit largement pour les calculer.

Ensuite, on regarde les termes en $x_i x_j$ avec $i \neq j$, sans oublier que $x_i x_j^2 = x_i x_j$. Cela donne $\frac{n(n-1)}{2}$ équations linéaires en $rn + n$ variables ν_{ki} et β_i.

Ainsi on récupère les ν_{ki} et donc la totalité des P_i. Nous avons montré qu'il suffit de pouvoir récupérer Q pour casser le Scotch.

10.1.1 Algorithme de récupération de Q.

1. On choisit un $t \neq 0$ et un $x^{(0)}$ au hasard.

2. On calcule $Q_{(r+1)..n}(x^{(0)}, t)$ qu'on notera $z_{(r+1)..n}$.

3. On cherche à résoudre en x l'équation

$$Q_{(r+1)..n}(x, t) = z_{(r+1)..n},$$

ce qui donnera au moins 2 solutions : $x^{(0)}$ et $x^{(0)} + t$ et on montre qu'il y a au plus $2 * 2^r$ solutions. Cela provient du fait que pour toute valeur de $z_{1..r}$ parmi 2^r possibles, l'équation $Q(x, t) = z$ a exactement 0 ou 2 solutions. Preuve :

$$Q(x,t) = P(x+t) - P(x) - P(t) = T(S^{1+2^\theta}(x+t) - S^{1+2^\theta}(x) - S^{1+2^\theta}(t)) =$$

$$T(S(x)S^{2^\theta}(t) + S(t)S^{2^\theta}(x))$$

(en utilise le fait que S et T sont linéaires)
De plus, S et T étant bijectives, il suffit de montrer que pour tout b et pour tout c, l'équation :

$$ac^{2^\theta} + a^{2^\theta}c = b$$

a exactement 0 ou 2 solutions en a. Supposons le contraire.
Il y a donc au moins une seule solution a. Il est facile à voir que $(a+c)$ en est une aussi. On a $c = S(t) \neq 0$, et il y a donc au moins 2 solutions distinctes $a \neq a'$. Cela donne :

$$ac^{2^\theta} + a^{2^\theta}c = a'c^{2^\theta} + a'^{2^\theta}c$$

On utilisera le fait que $x \mapsto x^{2^\theta}$ est linéaire.

$$(a-a')c^{2^\theta} + (a-a')^{2^\theta}c = 0,$$

donc comme $a \neq a'$ et $c \neq 0$ on a :

$$(a-a')^{2^\theta - 1} = c^{2^\theta - 1}$$

Or,

$$pgcd(2^\theta - 1, 2^n - 1) = pgcd(2^\theta - 1, 2^\theta) = 1,$$

la fonction $x \mapsto x^{2^\theta - 1}$ est donc bijective, et la seule solution possible est $(a - a') = c$. Comme $a + c + c = a$ cela donne que deux possibilités a et $(a+c)$.□

On ne retiendra que des valeurs de t pour lesquelles l'équation

$$Q_{(r+1)..n}(x,t) = z_{(r+1)..n},$$

a exactement $2 * 2^r$ solutions, sinon on revient au 1., et on recommence avec un couple $x^{(0)}$ et t différent. En pratique, l'espérance du temps de recherche reste à vérifier expérimentalement...

Tentative de justification : Il découle de ce qui précède, pour un t fixé, la probabilité qu'une valeur de y soit dans l'image de $x \mapsto Q(x,t)$ est $1/2$.

On supposera que l'ensemble image, appelé Z_t, des valeurs possibles de cette fonction, qui dépend de t, ressemble pour la majorité de t à un ensemble aléatoire. Soit

$$ZZ = \{z \in \mathbf{F}_{2^n} | z_{(r+1)..n} = Q_{(r+1)..n}(x^{(0)}, t)\}$$

l'ensemble de toutes les valeurs possibles que peut prendre $Q(x^{(0)}, t)$, en sachant que les bits $(r+1)..n$ sont donnés et avec les premiers r bits quelconques.

Le nombre des x qui satisfont l'équation

$$Q_{(r+1)..n}(x^{(0)}, t) = Q_{(r+1)..n}(x, t)$$

est alors égal à

$$2\#(Z_t \cap ZZ).$$

Or ZZ a 2^r éléments et donc la probabilité que $\#(Z_t \cap ZZ) = 2^r$ est de l'ordre de 2^{-r}, ce qui est une valeur non-négligeable, puisqu'on a supposé que 2^r est petit. □

4. Suite de l'attaque : Ainsi on trouve les $(2*2^r)$ solutions à l'équation $Q_{(r+1)..n}(x, t) = z_{(r+1)..n}$ qui vont par paires $(x, x + t)$ $(t \neq 0)$:

$$(x^{(0)}, x^{(0)} + t; x^{(1)}, x^{(1)} + t; \ldots; x^{(2^r-1)}, x^{(2^r-1)} + t).$$

Soit

$$U_i(x) = z_i$$

la i-ème équation manquante de $Q(x, t)$. Il est clair que $U_i : x \mapsto Q(x, t)$ est linéaire. En effet, on a vu que Q_ie s'écrivent comme $\sum \sim x_i tj$ et par conséquent, à t fixé elles s'écrivent comme $\sum \sim x_i$, sans terme affine.

Les $2 * 2^r$ solutions couvrent tous les cas de figure possibles pour les bits manquants, ce qui implique que pour la moitié des solutions le résultat $U_i(x) = z_i = 1$, et pour l'autre moitié $z_i = 0$.
Cela reste vrai si dans chacune des paires $(x, x + t)$ on choisit un seul élément, car la fonction $x \mapsto Q(x, t)$ prend la même valeur en x et $(x + t)$. Il est complètement indifférent lequel de 2 éléments on prend. On suppose que c'est $x^{(i)}$.

5. On somme toutes les équations $U_i(x) = y_i$:

$$\sum_i U_i(x_{(i)}) = 2^{r-1}$$

$$U_i(\sum_i x_{(i)}) = 2^{r-1}$$

Ceci donne une équation sur U_i.

Cette équation sur U_i est la même pour tout $i = 1..r$.

Si β_{ijk} est le coefficient de $x_j t_k$ dans l'équation $Q_i(x, t)$, on obtient une équation sur les β_{ijk} vraie $\forall i = 1..r$.

Dans une telle équation il y a $n(n+1)/2$ coefficients (encore les coefficients des x_i sont nuls).

6. Il faut répéter toute l'attaque 1.-5. $\mathcal{O}(n^2)$ fois, avec des $(x^{(0)}, t)$ différents, pour trouver le maximum possible d'équations indépendantes sur les β_{ijk} (elles ne dépendent pas de i). Le maximum a atteindre est $n(n+1)/2 - r$. Cela permettra d'obtenir ces r équations modulo une combinaison linéaire.
Ensuite on récupère les P_i manquantes, ou plus précisément leur espace vectoriel. □.

10.2 "Super Scotch"

Afin de contrer cette attaque et ses possibles généralisations nous recommandons de cacher dans le Scotch au moins 32 équations. Cela est appelé "Super Scotch"
L'attaque précédente est exponentielle en r, nombre d'équations manquantes, et on ne peut probablement pas l'améliorer dans ce sens.

Sixième partie

Autour du problème IP.

Chapitre 11

La résolution du problème IP.

Dans ce chapitre nous étudions différents attaques du problème général IP dans sa version de base avec 2 secrets, la même que nous avons introduite dans 4.1. On se restreint au cas où le nombre d'équations est égal au nombre d'inconnues $u = n$ et les fonctions S et T sont linéaires (au lieu des fonctions affines).
Rappelons son énoncé dans ce cas précis :

♣ **Données :** Soit $K = \mathbf{F}_q$ avec $q = p^m$, un corps fini.
Les lettres minuscules x dénotent habituellement des n-uplets $x = (x_1, \ldots, x_n)$ qui peuvent être vus, ou bien comme des éléments de K^n, ou bien comme étant dans $\mathbf{F}_{q^n}$.

$$\text{Soit } \mathcal{A} : \begin{cases} b_k = \sum_{i=0}^{n} \sum_{j=i}^{n} \lambda_{ijk} a_i a_j \\ \text{avec } k = 1..n \end{cases}$$

un ensemble de n équations quadratiques à n variables a_i (écrites avec la convention habituelle $a_0 = 1$).

$$\text{Soit } \mathcal{B} : \begin{cases} y_k = \sum_{i=0}^{n} \sum_{j=i}^{n} \lambda_{ijk} x_i x_j \\ \text{avec } k = 1..n \end{cases}$$

un autre ensemble d'équations quadratiques que l'on suppose isomorphe,i.e.

$$\mathcal{B} = T \circ \mathcal{A} \circ S$$

avec un changement de variables linéaire défini par deux endomorphismes inversibles $S : K^n \to K^n$ et $T : K^n \to K^n$:

$a = (a_1, \ldots, a_n) = S(x) = S(x_1, \ldots, x_n)$ et

$y = (y_1, \ldots, y_n) = T(b) = T(b_1, \ldots, b_n)$.

♣ **But :** Retrouver S et T étant donnés les deux ensembles d'équations $\mathcal{A}$ et $\mathcal{B}$.

Nous avons posé $u = n$ car c'est en quelque sorte le cas 'central'. En effet si $u = 1$ le problème est facile à résoudre (p.ex. voir chapitre 6 de [16]). De même pour $u > n^2/2$ dès que la famille des équations engendre l'ensemble de toutes les équations quadratiques le problème devient trivial, car tous ensembles d'équations deviennent isomorphes entre eux et pour tout S inversible on trouve facilement un T qui convient (ou plusieurs).

Le fait d'avoir séparé les cas linéaire et affine de IP demande de supposer que $\mathcal{A}(0) = 0$ et $\mathcal{B}(0) = 0$. Sinon on introduit une faiblesse artificielle au problème : on a $t(\mathcal{A}(0)) = \mathcal{B}(0)$ et cela donne un savoir $a\ priori$ sur t.

Les attaques sur IP que nous présenterons seront de complexité $q^{\mathcal{O}(n)}$. Les qualifier d'exponentielles est toutefois une affaire de point de vue, car la taille du secret dans IP est $\mathcal{O}(n^2)$, ce qui donne la recherche exhaustive en q^{n^2}. C'est beaucoup plus grand !

Par exemple pour une valeur réelle de $n = 80$ on compare 2^{40} à 2^{3200}.

Nos meilleures attaques permettent de résoudre IP en $n^{\mathcal{O}(1)}q^{n/2}$ avec $n^{\mathcal{O}(1)}q^{n/2}$ de mémoire.

La meilleure méthode connue à ce jour ([15], version étendue) était en $\mathcal{O}(q^{\sqrt{2}n\sqrt{n}})$. L'avance est considérable, car si encore on prend une valeur réelle de $n = 80$, on compare 2^{40} à 2^{360}.

Notre attaque comporte plusieurs variantes qui sont déstinées à traiter des différents cas particuliers. En effet, certaines fonctions très particulières pourraient ne pas satisfaire des hypothèses probabilistes en cours. Ainsi par exemple dans le cas de C^* dont la bijectivité rend le démarrage de l'attaque problématique, il a fallu concevoir une version particulière de l'attaque qui fonctionne dans ce cas.

Nous ne connaissons aucun cas de IP où cette attaque échoue dans son ensemble.

Un seul cas de IP a pu être résolu en temps polynomial : celui de D^* (voir 8.3). Par contre, il paraît difficile d'espérer de trouver une attaque générale, pour **tous** les IP, qui soit meilleure que $q^{n/2}$.

En principe les algorithmes sont probabilistes et supposent que la forme quadratique est choisie au hasard. Pour la description de certains algorithmes on va supposer qu'il existe une seule solution. Il est facile à justifier que la probabilité qu'il y ait plusieurs solutions est très faible, car une équation choisie au hasard n'a aucune raison de posséder une symétrie interne (un automorphisme non-trivial).

Le but des algorithmes est de trouver **une** solution et non pas toutes les solutions, bien que il semblerait que des fonctions qui admettent beaucoup d'automorphismes internes soient très rares et que la complexité de ce deux problèmes soit la même.

Nous avons trouvé plus de quatre méthodes générales pour résoudre IP.

11.1 Méthode du nombre d'antécédents.

Cette attaque utilise $\mathcal{O}(q^n)$ de mémoire.

L'idée de l'attaque consiste à identifier des ensembles de valeurs pour $\mathcal{A}$ et $\mathcal{B}$ ayant le même nombre d'antécédents.

Pour chacune des formes quadratiques $\mathcal{A}$ et $\mathcal{B}$ on utilise un tableau $\mathrm{H}_{\mathcal{A}}$ et $\mathrm{H}_{\mathcal{B}}$ de taille $\mathcal{O}(q^n)$ adressé par toutes les q^n valeurs de possibles à la sortie de $\mathcal{A}$ ou $\mathcal{B}$. Ce tableau sert à compter le nombre de fois qu'une valeur particulière est prise. On précalcule ce tableau pour $\mathcal{A}$ et $\mathcal{B}$ en $\mathcal{O}(q^n)$.

Ensuite on parcourt une deuxième fois toutes les valeurs de x pour chacune des formes quadratiques afin de séparer les entrées x en fonction du nombre d'antécédents de leur image $y = \mathcal{A}(x)$ ou $y = \mathcal{B}(x)$.

On note alors :

$\mathrm{G}_{\mathcal{A}}(x) \stackrel{def}{=} \mathrm{H}_{\mathcal{A}}(\mathcal{A}(x))$ est le nombre d'antécédents de $y = \mathcal{A}(x)$.

$\mathrm{G}_{\mathcal{A}}^{-1}(k) \stackrel{def}{=}$ l'ensemble des x tels que $y = \mathcal{A}(x)$ admet exactement k antécédents.

On n'a pas besoin de calculer le tableau $\mathrm{G}_{\mathcal{A}}$ en entier. On calcule directement tous les ensembles $\mathrm{G}_{\mathcal{A}}^{-1}(k)$ et $\mathrm{H}_{\mathcal{A}}^{-1}(k)$. pour $k = 1, 2$, etc. On n'a pas non plus besoin de les stocker, car dans la suite on n'utilisera que la somme (dans $\mathbf{F}_{q^n}$) des éléments de chacun des ensembles que l'on calcule itérativement.
On fait de même pour $\mathcal{B}$.

On peut évaluer précisément le nombre d'ensembles qu'on va obtenir.
On a vu dans 6.1. que pour une fonction (quadratique ou pas) choisie au hasard le nombre k d'antécédents d'une valeur suit la loi $P_k = \frac{1}{ek!}$. Cela implique que le nombre de valeurs dans $\mathrm{G}_{\mathcal{A}}^{-1}(k)$ descend à 1 vers :

$$k_{max},\ \mathrm{t}q.\ \frac{1}{ek_{max}!} = \frac{1}{q^n}$$

$$k_{max}! = q^n e^{-1}$$

On a $n! >> q^n$ pour $n >> q$ et $n! << q^n$ quand $n \leq q$. Ce qui donne une approximation :

$$k_{max} = \alpha n$$

où $\alpha < 1$ quand $n >> q$ et $\alpha > 1$ quand $n \leq q$.

Par l'isomorphisme des ensembles suivants sont en correspondance bijective :

$$\mathrm{G}_{\mathcal{A}}^{-1}(k) = S(\mathrm{G}_{\mathcal{B}}^{-1}(k)),\ pour\ k = 1, 2, \ldots, k_{max} \quad (1)$$

de même :

$$T\left(\mathrm{H}_{\mathcal{A}}^{-1}(k)\right) = \mathrm{H}_{\mathcal{B}}^{-1}(k),\ pour\ k = 1, 2, \ldots, k_{max} \quad (2)$$

L'attaque se base précisément sur ces relations. On cherchera à obtenir des équations de type $a = S(x)$ à partir de ces ensembles.

Soit Y, X deux ensembles. On notera $\hat{\sum}$ la somme d'éléments d'un ensemble. Si $Y = S(X)$ avec S linéaire. Alors

$$\hat{\sum} Y = S(\hat{\sum} X).$$

Ce qui permet d'obtenir directement à partir de (1) et (2), environ k_{max} équations sur S et autant sur T :

$$\begin{cases} S(x^{(1)}) = a^{(1)} \\ \vdots \\ S(x^{(k_{max})}) = a^{(k_{max})} \\ \\ T(b^{(1)}) = y^{(1)} \\ \vdots \\ T(b^{(k_{max})}) = y^{(k_{max})} \end{cases}$$

Et donc si $n \leq q$ on aura $k_{max} > n$ équations sur S et T ce qui est suffisant pour récupérer S et T par la réduction de Gauss. Dans le cas contraire, il faut d'avantage d'équations.

On peut envisager plusieurs méthodes pour en obtenir.

Dans la première, on divise K^n dans d'avantage de classes disjointes isomorphes pour $\mathcal{A}$ et $\mathcal{B}$ par l'intersection de plusieurs critères :

Raffinement Jusque là, nous avons classé les éléments à a ou x à l'entrée, respectivement de $\mathcal{A}$ ou $\mathcal{B}$, selon la valeur de $\mathrm{G}_{\mathcal{A}}(a)$ respectivement $\mathrm{G}_{\mathcal{B}}(x)$. Cela permet d'obtenir des classes qui sont en correspondance par S.

Avec **une seule** équation connue $S(x^{(1)}) = a^{(1)}$ on peut obtenir un découpage beaucoup plus fin en exigeant aussi que les valeurs $\mathrm{G}_{\mathcal{A}}(x + a^{(1)})$ et $\mathrm{G}_{\mathcal{B}}(a + x^{(1)})$ soient identiques dans une même classe, ce qui donnera d'avantage de classes. (!)

Ceci n'est pas la seule façon possible d'affiner les partitions.
Une alternative au nombre d'antécédents est l'étude des corrélations linéaires qui constitue à elle seule une méthode d'attaque très générale. Nous en décrirons une variante particulière adaptée à la situation.

11.2 Méthode des Corrélations Linéaires.

Cette méthode, comme la précédente, cherche à trouver les équations de type $a = S(x)$ en identifiant des comportements spécifiques de $\mathcal{A}$ et $\mathcal{B}$ qui donnent des ensembles d'entrées ayant le même comportement.

On va obtenir (ou affiner) un classement de x en le regardant comme un masque en entrée et en étudiant comportement des sorties en fonction de valeurs sur le masque. On peut également utiliser un masque à la sortie, ou prendre la moyenne sur des masques aléatoires. Ici on ne prendra pas de masque à la sortie mais on va analyser le "type" de la valeur obtenue à la sortie, en occurrence son nombre d'antécédents.

Le principe général de l'attaque : Pour deux masques : a à l'entrée de $\mathcal{A}$ et respectivement x à l'entrée de $\mathcal{B}$ correspondants (i.e. $a = S(x)$) on obtient des statistiques sur le comportement des sorties identiques.

Soit $x \in K^n$ un masque. On regarde en fonction d'un $x' \in K^n$, la corrélation entre le nombre k d'antécédents de $y' = \mathcal{B}(x')$ et la valeur de la forme linéaire définie par x qui est $\sum_{i=1}^{n} x_i x'_i$ (compter les x' qui donnent chacune de valeurs de k possible). Pour tout x, on fait à priori une recherche exhaustive sur x', en pratique on peut se contenter des classes moins précises.
Nous ne décrirons pas cette approche en détail, car elle donne les résultats comparables à l'attaque des antécédents avec raffinements, elle est beaucoup plus complexe à mettre en oeuvre et nécessite d'avantage de mémoire.
Elle est également beaucoup moins prévisible.
Il est possible toutefois, qu'elle fonctionne dans certains cas où la méthode précédente échoue, mais elle échoue dans le cas un C^*.

11.3 La méthode de va-et-vient.

Les deux façons décrites précédemment d'améliorer l'attaque sont superflues, car nous allons montrer qu'étant donné deux, et dans certains cas même une, équation (!!!) connue sur S ou sur T on peut en récupérer d'avantage (autant qu'on veut). Dans certains cas, si le nombre d'équations au départ est petit, la méthode exige une puissance de calcul de q^n **ou** une attaque à chiffré choisi. Dans le cas où on dispose dès le début de plusieurs, environ $\mathcal{O}(\log n)$ équations sur S, elle est polynomiale (!).
Cette méthode est appelée "va-et-vient". Elle s'affranchit totalement de toutes les hypothèses statistiques et fonctionne pour toute fonction cryptographique cachée par S et T, même si son degré est > 2.

Le lemme de va-et-vient.

Avec k équations connues sur S (de type $a = S(x)$) on peut obtenir en temps polynomial environ $q^k - 1$ équations sur T.

Avec k équations connues sur T on peut obtenir environ $q^k - 1$ équations sur S, moyennant la possibilité de calculer $\mathcal{A}^{-1}$ et $\mathcal{B}^{-1}$, avec $\mathcal{O}(q^n)$ calculs ou la connaissance de la forme interne $\mathcal{A}$ avec l'attaque à texte chiffré choisi sur $\mathcal{B}$.

En pratique, si on veut améliorer les 2 attaques décrites précédemment, il suffira d'un seul passage par le va-et-vient dans le sens direct, sans les inversions de $\mathcal{A}$ et $\mathcal{B}$. Cela est dû au fait que $k_{max} = \alpha n$ et $q^{k_{max}}$ devrait être très grand. Le va et vient sera polynomial.

Preuve du lemme :

D'abord la première partie (sens direct).

Étant donné k équations :

$$\begin{cases} S(x^{(1)}) &= a^{(1)} \\ & \vdots \\ S(x^{(k)}) &= a^{(k_{max})} \end{cases}$$

linéairement indépendantes, on peut obtenir $q^k - 1$ équations dépendantes :

$$\begin{cases} S(\sum \sim x^{(i)}) &= \sum \sim a^{(i)} \\ & \vdots \end{cases}$$

Ces équations, bien que dépendantes, donneront des équations indépendantes, une fois "transférées" de l'autre côté. C'est surprenant !

En effet, on y applique $\mathcal{A}$, qui n'a aucune raison d'être linéaire. Si c'est une fonction difficile à inverser, elle est fortement non-linéaire la plupart de temps.

Les équations :

$$\begin{cases} \mathcal{A}\left(S(\sum \sim x^{(i)})\right) &= \mathcal{A}\left(\sum \sim a^{(i)}\right) \\ & \vdots \end{cases}$$

Seront donc indépendantes pour la plupart.

Ensuite on applique T (formellement) :

$$\begin{cases} T\left(\mathcal{A}(S(\sum \sim x^{(i)}))\right) &= T\left(\mathcal{A}(\sum \sim a^{(i)})\right) \\ & \vdots \end{cases}$$

ce qui, en utilisant le forme publique $\mathcal{B} = T \circ \mathcal{A} \circ S$ donne environ $q^k - 1$ équations sur T qui restent indépendantes :

$$\begin{cases} \mathcal{B}(\sum \sim x^{(i)}) &= T\left(\mathcal{A}(\sum \sim a^{(i)})\right) \\ & \vdots \end{cases}$$

□

Pour la preuve dans l'autre sens, on part des équations :

$$\begin{cases} T(b^{(1)}) &= y^{(1)} \\ & \vdots \\ T(b^{(k)}) &= y^{(k_{max})} \end{cases}$$

On écrit des combinaisons linéaires :

$$\begin{cases} T(\sum \sim b^{(i)}) &= \sum \sim y^{(i)} \\ & \vdots \end{cases}$$

On applique $\mathcal{B}^{-1}$: Comme il n'est pas bijectif, on obtient des égalités ensemblistes sur des (petits) ensembles) :

$$\begin{cases} \mathcal{B}^{-1}\left(T(\sum \sim b^{(i)})\right) &= \mathcal{B}^{-1}\left(\sum \sim y^{(i)}\right) \\ & \vdots \end{cases}$$

Puis on applique S :

$$\begin{cases} S\left(\mathcal{B}^{-1}\left(T(\sum \sim b^{(i)})\right)\right) &= S\left(\mathcal{B}^{-1}\left(\sum \sim y^{(i)}\right)\right) \\ & \vdots \end{cases}$$

Maintenant on utilise le fait que

$$\mathcal{B} = T \circ \mathcal{A} \circ S$$

ce qui donne une équation ensembliste :

$$\mathcal{B}^{-1} = S^{-1} \circ \mathcal{A}^{-1} \circ T^{-1}$$

$$S \circ \mathcal{B}^{-1} \circ T = \mathcal{A}^{-1}$$

On obtient environ $q^k - 1$ équations ensemblistes sur S :

$$\begin{cases} \mathcal{A}^{-1}\left(\sum \sim b^{(i)}\right) & = & S\left(\mathcal{B}^{-1}(\sum \sim y^{(i)})\right) \\ & & \vdots \end{cases}$$

auxquelles on applique l'opérateur $\hat{\sum}$ de la somme d'éléments d'un ensemble :

$$\begin{cases} \hat{\sum}\left(\mathcal{A}^{-1}(\sum \sim b^{(i)})\right) & = & S\left(\hat{\sum}(\mathcal{B}^{-1}(\sum \sim y^{(i)}))\right) \\ & & \vdots \end{cases}$$

Ainsi nous avons obtenu environ $q^k - 1$ équations explicites sur S.

□

On remarque que si $q > 2$, en moyennant une puissance de calcul globale de $\mathcal{O}(\log nq^n)$ une seule (!) équation de type $a = S(x)$ suffit pour "amorcer" le va-et-vient car $q^k - 1$ sera ≥ 3. L'inversion des formes $\mathcal{A}$ et $\mathcal{B}$ est nécessaire $\mathcal{O}(\log n)$ fois seulement. En effet, ce nombre d'équations sur S suffit pour récupérer n équations sur T par le va-et-vient direct, et donne T par une réduction de Gauss.
Les inversions n'augmentent donc pas la complexité de façon significative qui est de $\mathcal{O}(q^n)$ pour trouver la première équation. Les précalculer sous forme d'un tableau prendrait beaucoup trop de mémoire.

À titre d'illustration nous allons attaquer un exemple concret, dans un des cas difficiles, un C^*, qui résiste aux autres attaques.

11.4 Application : résolution d'un C*

Seule la méthode de "va-et-vient" (11.3) (avec recherche exhaustive sur 2 équations) permet de résoudre ce cas. Nous avons implémenté les méthodes "des antécédents", et "des corrélations". Nous avons constaté qu'elles échouent à obtenir la moindre information sur S ou T. Ceci est probablement du au fait qu'il existe plusieurs solutions qui donnent trop de symétries aux équations.
L'attaque de IP "Unique Source" décrite plus tard ne traite pas ce cas.

En effet, l'application $x \mapsto x^{1+2^\theta}$ de $\mathbf{F}_2^n \to \mathbf{F}_2^n$ admet $n(2^n-1)$ automorphismes sous la forme

$$\begin{cases} S : x \mapsto \alpha x^{2^\beta} \\ T : x \mapsto \alpha^{-2^\theta} x^{2^{-\beta}} \end{cases} \quad \text{avec } \alpha \neq 0,\ \beta \in \{0, 1, \ldots, n-1\}.$$

Cette multitude de solutions nous a permis de réduire la complexité de l'attaque de façon spectaculaire. En effet pour trouver 2 équations initiales sur S qui amorcent le "va-et-vient" on a besoin de 2^{2n} essais. Pour trouver une des solutions possibles, il en suffit $2^n/n$.

Ainsi l'attaque est en $n^{\mathcal{O}(1)}q^n$ et on utilise une petite quantité de mémoire de l'ordre de $\mathcal{O}(n^2)$.
La recherche exhaustive sur S (ou T, l'un donne l'autre) serait en $\mathcal{O}(2^{n^2})$, soit en $\mathcal{O}(2^{32})$.

On va prendre l'exemple qui apparaî t dans l'article [14], chapitre 11.4., en bas de la page 27 (version étendue). Ici $q = 2$, le nombre d'équations $n = 5$.

$$\mathcal{B} : \begin{cases} y_0 = x_0 + x_2 + x_3 + x_4 + x_0x_1 + x_0x_2 + x_0x_4 + x_1x_3 + x_2x_4 + x_3x_4 \\ y_1 = x_0x_3 + x_0x_4 + x_1x_2 + x_1x_3 + x_1x_4 + x_3x_4 \\ y_2 = x_0 + x_2 + x_3 + x_0x_1 + x_0x_4 + x_1x_2 + x_2x_3 \\ y_3 = x_2 + x_3 + x_0x_1 + x_0x_4 + x_2x_3 + x_2x_4 + x_3x_4 \\ y_4 = x_1 + x_3 + x_4 + x_0x_2 + x_0x_3 + x_1x_2 + x_1x_3 + x_1x_4 \end{cases}$$

On utilise le polynôme irréductible $z \mapsto z^5+z^2+1$ et on écrit l'expression de $x \mapsto x^3$:

$$\mathcal{A} : \begin{cases} b_0 = a_0 + a_0a_4 + a_1a_2 + a_1a_3 + a_2a_3 \\ b_1 = a_2 + a_3 + a_4 + a_0a_1 + a_0a_3 + a_3a_4 \\ b_2 = a_4 + a_0a_1 + a_0a_2 + a_0a_4 + a_1a_2 + a_2a_4 + a_3a_4 \\ b_3 = a_1 + a_2 + a_3 + a_4 + a_0a_4 + a_2a_3 + a_2a_4 \\ b_4 = a_3 + a_0a_2 + a_0a_4 + a_1a_2 + a_1a_3 + a_1a_4 + a_2a_3 + a_2a_4 + a_3a_4 \end{cases}$$

La programmation de va-et-vient nous a permis, en un rien de temps, de trouver **exactement** le nombre prévu de solutions, $155 = n(2^n-1)$.

L'attaque est tellement simple, qu'on va la refaire sans ordinateur. On va ainsi obtenir une de ces 155 solutions.

On va coder les éléments de $\mathbf{F}_2^n$ par des entiers dont le coefficient de 2^i en écriture binaire représente x_i. Cela permet d'écrire un tableau de valeurs de fonctions $\mathcal{A}$ et $\mathcal{B}$. Par exemple dans la colonne 7 on lit $\mathcal{A}(5) = 31$ ce qui signifie que si $a = (a_4, a_3, a_2, a_1, a_0) = (0, 0, 1, 0, 1)$ alors $x = S(a)$ vaut $(1, 1, 1, 1, 1)$.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$\mathcal{A}$	0	1	8	15	10	31	23	4	26	25	3	6	9	30	5	20	14	18	22	12	24	16	21	27	2	28	11	19	13	7	17	29
$\mathcal{B}$	0	5	16	24	13	25	11	18	29	10	30	4	28	26	9	2	17	27	19	20	21	14	1	23	7	31	22	3	15	6	8	12

On a une chance sur $2^{10}/155 \sim 6.6$ environ de trouver une bonne valeur initiale pour 2 équations. On va commencer par les suivantes :

$$\begin{cases} S(1) = 1 \\ S(2) = 7 \end{cases}$$

Ce qui donne 3 équations linéairement dépendantes sur S :

$$\begin{cases} S(1) = 1 \\ S(2) = 7 \\ S(3) = 6 \end{cases}$$

En effet par la linéarité $S(1+2) = 1+7$, les calculs se font dans $\mathbf{F}_{32}$, et cela donne $S(3) = 6$.

A l'aide du tableau de valeurs de $\mathcal{A}$ et $\mathcal{B}$ on obtient 3 équations indépendantes sur T :

$$\begin{cases} T(1) = 5 \\ T(4) = 16 \\ T(23) = 24 \end{cases}$$

Par exemple en partant de $S(2) = 7$, on calcule $\mathcal{A}(7) = 4$ et $\mathcal{B}(2) = 16$ ce qui permet de déduire que $T(4) = 16$.

Par combinaisons linéaires de ces équations on obtient $2^3 - 1 = 7$ équations dépendantes :

$$\begin{cases} T(1) = 5 \\ T(4) = 16 \\ T(5) = 21 \\ T(18) = 13 \\ T(19) = 8 \\ T(22) = 29 \\ T(23) = 24 \end{cases}$$

Et cela permet, par l'utilisation de $\mathcal{A}^{-1}$ et $\mathcal{B}^{-1}$ de déduire 7 équations sur S :

$$\begin{cases} S(1) = 1 \\ S(2) = 7 \\ S(3) = 6 \\ S(4) = 17 \\ S(8) = 18 \\ S(20) = 14 \\ S(30) = 27 \end{cases}$$

Parmi ces 7 équations 6 sont indépendantes, ce qui suffit pour retrouver S par réduction de Gauss. On récupère de même façon T et on vérifie que la solution est correcte.

Voilà la solution que nous avons obtenue :

$$(a_0, a_1, a_2, a_3, a_4) = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix} (x_0, x_1, x_2, x_3, x_4)$$

$$(y_0, y_1, y_2, y_3, y_4) = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \end{bmatrix} (b_0, b_1, b_2, b_3, b_4)$$

□

11.5 Améliorations du "va-et-vient".

On cherche un façon d'améliorer le va-et-vient pour commencer avec une seule équation, ce qui (dans le va-et-vient) pose problème particulier pour $q = 2$. On aimerait aussi disposer d'une version de va-et-vient qui ne nécessite pas d'inversion des formes quadratiques $\mathcal{A}$ ou $\mathcal{B}$ et qui permettrait d'imaginer les attaques en moins que q^n. Pour cela il serait souhaitable de disposer d'une fonction appelée fonction de gain, appelée F, qui aurait les propriétés suivantes :

Étant donné une entrée x on obtient avec une probabilité non-négligeable $P \in \mathcal{O}(1)$ et en temps polynomial, une valeur $F(x) = x'$ telle que la fonction F est préservée par l'isomorphisme. C'est à dire :

Si $S(x) = a$, $F_{\mathcal{B}}(x) = x'$ et $F_{\mathcal{A}}(a) = a'$

Alors $S(x') = a'$.

Remarque : dans cette définition on peut supprimer la partie "avec une probabilité non négligeable" si l'on pose $F(x) = 0$ à chaque fois que la fonction F est non définie. (La valeur 0 est toujours préservée par une application linéaire.)

Trouver une telle fonction n'est pas facile.

Théorème : Il n'existe pas de fonction de gain non triviale qui fonctionne pour tous les IP.
Preuve : On a vu dans 11.4. que la connaissance de 2 équations sur S permet, par le va-et-vient de récupérer S et T de façon déterministe (du moins dans certains cas). Une fonction de gain permettrait dans certains cas d'obtenir 2 équations à partir d'une seule. Or, connaissant la structure des isomorphismes de C^* qui est décrite dans 11.4., on montre facilement qu'il existe exactement n isomorphismes qui satisfont une équation donnée sur S $\square$.

La définition d'une fonction de gain qu'on a pris est trop restrictive.

Nous chercherons des fonctions plus générales qui manipulent des ensembles d'information non seulement à l'entrée mais aussi à la sortie de la fonction $\mathcal{A}$ ou $\mathcal{B}$. S'il s'agit seulement de l'information de type valeurs à l'entrée ou à la sortie qui se correspondent par l'isomorphisme, la démonstration de non-existence de telles fonction reste valable. Par contre on verra dans 11.7.4. qu'il existe de telles fonctions de gain pour des ensembles d'information très différents.

Indépendamment de cela, nous allons "libéraliser" les axiomes d'une fonction de gain dans d'autres directions :
-on exige que le résultat soit correct seulement avec une certaine probabilité non-négligeable.
-on peut utiliser dans une même attaque plusieurs fonctions de gain selon les caractéristiques de la forme quadratique.
-on utilise des fonctions probabilistes (non-déterministes).
-enfin, on exige que les données obtenues soient préservées non plus par tout isomorphisme, mais par au moins un isomorphisme entre $\mathcal{A}$ et $\mathcal{B}$.

C'est précisément les deux dernières conditions qui infirment la démonstration de non-existence des fonctions de gain.

Une telle fonction sera appelée une fonction de gain faible et notée F. Par abus de langage on l'appellera encore une fonction de gain.

Nous avons trouvé des nombreuses fonctions de gain plus ou moins complexes.

Pour la caractéristique du corps de base $p \neq 2$, la plupart de temps la fonction suivante fonctionne :

$$E(z) = \mathcal{B}(rz), \text{ avec } r \in \mathbf{F}_p$$

et donne plusieurs équations sur T à partir d'une seule équation sur S.

Même pour $p \neq 2$ la fonction précédente est peu commode car elle donne de l'information sur T et pas sur S. Nous chercherons d'autres fonctions de gain.

Soit z une donnée initiale de la fonction $\mathcal{B}$.
On commence par chercher tous les x tels que

$$\mathcal{B}(x+z) = \mathcal{B}(x).$$

Cette recherche se fait en temps polynomial en utilisant la forme polaire de $\mathcal{B}$ comme décrit dans 5.3.

Malheureusement on trouve toujours plusieurs valeurs x. On peut vouloir prendre leur somme, mais cela pose un problème singulier en caractéristique $p = 2$: Si x est une solution $x + z$ en est une aussi, et la somme de deux solutions est z. On ne trouve ainsi aucune nouvelle valeur.

On va garder parmi x et $x + z$ qu'une seule valeur choisie au hasard. Ainsi la somme de toutes les valeurs sera exacte dans le cas $p \neq 2$ et sera connue à $+z$ près, donc avec une probabilité de $1/2$, si p=2. Cela constitue notre fonction de gain :

$$F_{\mathcal{B}}(z) = \sum_{\substack{x \text{ tels que} \\ \mathcal{B}(x+z)=\mathcal{B}(x) \\ \text{un seul parmi } \{x,x+z\}}} x.$$

On la définit de même pour $\mathcal{A}$:

$$F_{\mathcal{A}}(c) = \sum_{\substack{a \text{ tels que} \\ \mathcal{A}(a+c)=\mathcal{B}(a) \\ \text{un seul parmi } \{a,a+c\}}} a.$$

Cette fonction ne fonctionne évidemment que dans les cas non-bijectifs. Pour des fonctions bijectives on utilise une autre fonction de gain définie comme suit :

$$G_{\mathcal{B}}(z) = \sum_{\substack{x \text{ tels que} \\ \mathcal{B}(x+z)=\mathcal{B}(x)+\mathcal{B}(z) \\ \text{un seul parmi } \{x,x+z\}}} x.$$

Le principe de résolution est le même que précédemment, et on définit une fonction analogue pour $\mathcal{A}$.

Dans le cas de C^*, vu le théorème de la page précédente, les deux fonctions F et G ne peuvent qu'échouer. On s'en sort pourtant facilement en modifiant légèrement la fonction F. On pose :

$$H_{\mathcal{B}}(z) = \sum_{\substack{x \text{ tels que} \\ \mathcal{B}(x+z)=\mathcal{B}(x)+\alpha_{\mathcal{B}} \\ \text{un seul parmi } \{x,x+z\}}} x.$$

où $\alpha_{\mathcal{B}}$ est une valeur quelconque, choisie au hasard en fonction de $\mathcal{B}$ et toujours la même pour le même $\mathcal{B}$. On fait de même pour $\mathcal{A}$.

On pourrait également modifier G de même façon.

Le fonctionnement de H est un peu surprenant. Ce qui gêne c'est le nombre de solutions trop grand pour avoir une fonction de gain 'universelle', c'est à dire qui est conservée par tout S et T. La solution est très simple et consiste a restreindre le nombre de solutions S et T car l'attaque ne trouvera que celles où $T(\alpha_{\mathcal{A}}) = \alpha_{\mathcal{B}}$.

Dans le chapitre 11.7.3 nous décrivons une autre fonction de gain possible K, qui peut servir s'il s'avère que F, G et H échouent dans un cas très particulier.

11.6 Attaque "Va-et-vient dans le milieu".

Nous décrirons une attaque conçue pour résoudre **tous** les cas de IP avec la complexité $n^{\mathcal{O}(1)}q^{n/2}$ et avec $n^{\mathcal{O}(1)}q^{n/2}$ de mémoire.

C'est une version du va-et-vient utilisant le paradoxe d'anniversaires pour diminuer la complexité. Elle utilise les fonctions de gain décrites précédemment. Il va de soi, comme pour le majorité des attaques basées sur le paradoxe d'anniversaires, qu'elle peut facilement être transformée en une attaque en $n^{\mathcal{O}(1)}q^{n}$ avec $n^{\mathcal{O}(1)}$ de mémoire. D'autres arrangements sont possibles en fonction de la mémoire disponible. Nous décrirons la version la moins gourmande en calcul en $n^{\mathcal{O}(1)}q^{n/2}$.

1. Début.

On choisi au hasard $(\log n)^{\mathcal{O}(1)}q^{n/2}$ valeurs $z^{(i)}$ et $(\log n)^{\mathcal{O}(1)}q^{n/2}$ valeurs $c^{(i)}$ (on peut commencer avec les mêmes). Ainsi par le paradoxe d'anniversaires il existe au moins $(\log n)^{\mathcal{O}(1)}$ collisions telles que $S(z^{(i)}) = c^{(j)}$.

2. Démarrage (1 valeur $\mapsto \log\log n + \mathcal{O}(1)$ valeurs) :

On pose $z^{(i,0)} = z^{(i)}$ et $c^{(i,0)} = c^{(i)}$. On applique successivement $\log\log n + \mathcal{O}(1)$ fois la fonction de gain adaptée :

$$z^{(i,j+1)} = F_{\mathcal{B}}(z^{(i,j)})$$

et

$$c^{(i,j+1)} = F_{\mathcal{A}}(c^{(i,j)})$$

(éventuellement remplacer F par F, G, H ou K.)

Dans cette étape on va écarter tous les cas où une certaine valeur $F_{\mathcal{B}}(z^{(i,j)})$ ou $F_{\mathcal{A}}(c^{(i,j)})$ est nulle. Dans ce cas on jette les valeurs initiales $z^{(i)}$ ou $c^{(i)}$ correspondantes. Ainsi gardera 1 valeur sur environ $\mathcal{O}(1)^{\log\log n+\mathcal{O}(1)} = (\log n)^{\mathcal{O}(1)}$.

Pour $p = 2$, sachant qu'un aléa avait été introduit dans les fonctions de gain, il y a plusieurs suites $(z^{(i,j)}, j = 1..)$ possibles. Il reste que le nombre de possibilités est assez restreint : $2^{\log\log n+\mathcal{O}(1)} = \mathcal{O}(\log n)$.

Ayant rejeté un certain nombre de valeurs initiales, on peut toujours supposer qu'il reste $\mathcal{O}(\log n)$ collisions $S(z^{(i)}) = c^{(j)}$. Et parmi ces collisions au moins $\mathcal{O}(1)$ ont fait les choix correspondants dans toutes les applications de la fonction probabiliste F, G, H, c'est à dire qu'il existe au moins $\mathcal{O}(1)$ paires $(z^{(i)}, c^{(j)})$ telles que

$$\forall k = 1..(\log\log n + \mathcal{O}(1)), \quad S(z^{(i,k)}) = c^{(j,k)}$$

3. Pré-Expansion ($\log\log n + \mathcal{O}(1)$ valeurs à l'entrée $\mapsto \log n + \mathcal{O}(1)$ valeurs à la sortie) :

On calcule $\mathcal{B}$ appliqué à toutes les combinaisons linéaires des $z^{(i)}$ qui sont en nombre $q^{\log\log n + \mathcal{O}(1)} > \log n + \mathcal{O}(1)$.

On note $t^{(i,j)}, j = 1..(\log n + \mathcal{O}(1))$ les valeurs obtenues. On peut supposer qu'au moins $\log n + \mathcal{O}(1)$ sont linéairement indépendantes, et on fait des opérations de façon à écrire toujours la suite $t^{(i,j)}, j = 1..(\log n + \mathcal{O}(1))$ dans le même ordre.
On répète exactement les mêmes opérations pour $\mathcal{A}$ et on obtient les $d^{(i,j)}$.

4. Expansion ($\log n + \mathcal{O}(1)$ valeurs à la sortie $\mapsto n + \mathcal{O}(1)$ valeurs à l'entrée) : Pour tout candidat i on fera encore la 'résolution différentielle' de $\mathcal{B}$ où la différence à l'entrée sera toujours la même $z^{(i,0)}$, et les différences à la sortie seront **toutes** les combinaisons linéaires possibles des valeurs $t^{(i,j)}, j = 1..(\log n + \mathcal{O}(1))$. On obtient ainsi $q^{\log + \mathcal{O}(1)} > n + \mathcal{O}(1)$ valeurs définies comme suit :

$$\sum_{\substack{\text{les } x \text{ tels que} \\ \mathcal{B}(x+z)-\mathcal{B}(x)=\sum_j \sim t^{(i,j)} \\ \text{un seul parmi } \{x, x+z\}}} x.$$

où $\sim$ sont des coefficients quelconques dans $\mathbf{F}_q$.

Parmi ces nombres on retient $n + \epsilon$, $\epsilon \in \mathcal{O}(1)$ qu'on écrit encore dans un ordre fixé, et qu'on nomme $z^{(i,j)}, j = 1..(n + \epsilon)$.

On procède de façon analogue pour $\mathcal{A}$ et on obtient les $c^{(i,j)}, j = 1..(n + \epsilon)$.

5. Étiquettage.

Pour chacune des suites $z^{(i,j)}, j = 0..(n + \epsilon)$ on détecte toutes les relations de dépendance linéaire possibles entre chacun des $z^{(i,j)}$ et tous les $z^{(i,k)}, k < j$ qui sont non nuls et n'ont pas été trouvés en dépendance linéaire avant. Ainsi pour la suite donnée on obtient un ensemble, ayant au moins ϵ membres, des équations de dépendance linéaire de la suite. Ceci constitue une étiquette, une sorte de signature unique d'une suite.

6. Détection des collisions.

Supposons d'abord que $p \neq 2$.

Si deux suites, une pour $\mathcal{A}$ et l'autre pour $\mathcal{B}$ sont en dépendance linéaire, elles auront la même étiquette. Si ce n'est pas le cas, la probabilité qu'elles aient la même étiquette est négligeable et peut être rendue aussi petite qu'on veut : $1/q^{n\epsilon}$. On voit que $\epsilon = 2$ est suffisant.

Si $p = 2$ dans deux suites qui correspondent par S ont le défaut supplémentaire d'être connues à $(+z^{(i,0)})$ près. Cela peut toujours être détecté grâce au fait que pour un i fixé c'est toujours le même $z^{(i,0)}$. En effet, la probabilité qu'un suite ayant $n+\epsilon$ éléments et la même suite où à certain éléments choisi au hasard on ajoute une même valeur fixée $z^{(i,0)}$ admettent **exactement les mêmes** relations linéaires est relativement grande : $\frac{1}{2^\epsilon}$. Cela montre que si on procède exactement comme pour $p \neq 2$, avec la probabilité $\frac{1}{2^{2\epsilon}}$ on détectera quand même deux valeurs $z^{(i,0)}$ et $c^{(j,0)}$ qui se correspondent par S. On posera encore $\epsilon = 2$, ce qui donne $\frac{1}{2^{2\epsilon}} = \frac{1}{16}$. Si $p = 2$ une valeur de S trouvée par l'attaque a une probabilité $\frac{1}{16}$ d'être correcte.

Donc dans tous les cas on cherche deux indices i et j tels que les suites, $z^{(i,k)}, k = 0..(n+\epsilon)$ et $c^{(j,k)}, k = 0..(n+\epsilon)$ ont la même étiquette.

7. Calcul de S.

Cela permet immédiatement de trouver S par réduction de Gauss sur $> n$ valeurs qu'on sait ainsi relier.

8. Fin de l'attaque.

Avec S il est facile d'obtenir T et on vérifie si la solution obtenue est correcte. Éventuellement, si $p = 2$, il faut réessayer environ 16 fois.

□

Nous ne connaissons aucun cas particulier de forme quadratique, ou toutes les trois variantes (avec F, G et H) de l'attaque présentée échouent dans l'ensemble.

11.7 Propriétés avancées de IP.

L'attaque du va-et-vient et ces améliorations exploitent le lien entre l'information sur S et l'information sur T, avec des propriétés surprenantes de type gain d'information :

♣ La connaissance de $\log n$ lignes de matrice S (ou leur combinaisons linéaires) permet de connaî tre en temps polynomial T (en totalité).

♣ La connaissance de 2 (une si $p \neq 2$) lignes (ou leur combinaisons linéaires) de n'importe quelle matrice S ou T permet de tout récupérer moyennant l'inversion de $\mathcal{A}$ et $\mathcal{B}$.
La fonctions de gain de l'étape 2. de l'attaque "va-et-vient dans le milieu" donnent des moyens pour ce faire sans l'inversion des formes quadratiques.

Il n'y pas de certitude qu'une forme très particulière résiste à toutes ces fonctions, et pour cela, et dans le cas $p = 2$, nous présenterons un procédé différent de "gain d'information", dit de "relations conjuguées".

Il est de nature un peu différente, car il lie une information sur des lignes de la matrice S avec une information sur des colonnes de la matrice T. Il ne peut donc pas être directement "emboî té" avec les transformations du va-et-vient. Il permet de :

♣ Étant donné une équation sur S (i.e. connaissant une combinaison linéaire des lignes de S) on peut obtenir avec la probabilité $p_{\#} \sim 0.3$ une combinaison linéaire des colonnes de T.

♣ On peut également procéder de façon inverse (avec une certaine probabilité à évaluer).

♣ Le lien entre l'information "de deux côtés" est bijectif et inversible, mais **non linéaire**, ce qui permet de faire apparaî tre des phénomènes de "gain d'information" comme dans la va-et-vient qui transforment des équations linéairement dépendantes en équations linéairement indépendantes.

♣ La structure de ce procédé permet encore de l'utiliser dans les algorithmes de type "attaque du milieu".

11.7.1 Principe de relations conjuguées.

On va supposer que les formes quadratiques $\mathcal{A}$ et $\mathcal{B}$ ont très peu de caractère linéaire (sinon elles auraient peu d'utilité en cryptographie).

On va commencer avec une valeur initiale x pour $\mathcal{B}$. Soit a une valeur initiale pour $\mathcal{B}$. Pour l'instant on prend x et a quelconques.

On cherchera encore des solutions à l'équation :

$$\mathcal{A}(x + z) = \mathcal{A}(x)$$

.

Le principe des "relations conjuguées" s'applique dans les cas où cette équation admet **exactement** deux solutions x et $x + z$. On va évaluer la probabilité $p_{\#}$ de ce cas.

On suppose que la forme $\mathcal{A}$ avait été choisie au hasard. Soit p_i la probabilité qu'une valeur y choisie au hasard ait i antécédents. D'après 6.1. on a $p_i = \frac{1}{ei!}$.

Une valeur y ayant i antécédents, ce qui arrive avec la probabilité p_i, produit $\binom{i}{2}$ différences à l'entrée. (tous les couples possibles x et x' tq. $\mathcal{A}(x) = \mathcal{A}(x')$).

Pour l'instant on ne tiendra pas compte de collisions possibles et l'on considère des ensembles avec répétition. Ainsi on obtient

$$2^n \sum_{i=2}^{\infty} \binom{i}{2} \frac{1}{ei!} = \frac{1}{2} 2^n$$

différences z avec des répétitions, dont celles qui nous intéressent sont celles qui ne se répètent pas, car c'est elles qui font que l'équation $\mathcal{A}(x+z) = \mathcal{A}(x)$ aura exactement deux solutions.

Le modèle probabiliste qu'on va utiliser est celui de tirage aléatoire de $2^n/2$ points parmi 2^n, ce qui revient à supposer que les différences d'entrées correspondantes à des différents y, et à des différents choix des antécédents x et x‘ de y, sont toutes indépendantes. La probabilité qu'un point soit choisi exactement une fois est :

$$p_{\#} = \sum_1^{2^n/2} \frac{1}{2^n} (1 - \frac{1}{2^n})^{2^n/2-1} \sim \frac{1}{2} \sqrt{\frac{1}{e}} \sim 0.30.$$

Ainsi la probabilité que l'équation $\mathcal{A}(x+z) = \mathcal{A}(x)$ ait exactement 2 solutions est de l'ordre de $p_{\#} = 0.3$. Pour la variante 2^{III} décrite plus loin de l'attaque "va-et-vient dans le milieu" on fera seulement l'hypothèse que $p_{\#} \in \mathcal{O}(1)$. Nous ne connaissons aucun cas particulier où cette probabilité soit trop petite, ce qui empêcherait ces attaques, sauf dans les cas bijectifs où $p_{\#} = O$.

Le principe de relations conjuguées.

Si l'équation $\mathcal{B}(x+z) - \mathcal{B}(x) = 0$ admet exactement 2 solutions en les x_i pour un z donné, les n équations

$$\mathcal{B}_i(x+z) - \mathcal{B}_i(x), \ i = 1..n$$

dans lesquelles on néglige les termes constants forment un système de rang $n-1$.
On montre facilement que ces termes constants valent $\mathcal{B}_i(z), \ i = 1..n$ et donc les equations de la forme polaire de $\mathcal{B}$:

$$y_i = \mathcal{B}_i(x+z) - \mathcal{B}_i(x) - \mathcal{B}_i(z), \ i = 1..n$$

admettent une unique relation non nulle de type

$$\sum \sim y_i = 0.$$

Cette relation liée à z est conservé par l'isomorphisme : si $\mathcal{B} = T \circ \mathcal{A} \circ S$ et si $S(z) = c$ alors les deux relations non nulles $\sum \sim y_i = 0$ et $\sum \sim b_i = 0$ obtenues comme plus haut, sont identiques modulo le changement de variables induit par T :

$$y = T(b).$$

Réciproquement, supposons que l'on dispose de deux relations isomorphes liées par T

$$\sum \sim y_i = 0 \textbf{ et } \sum \sim b_i = 0 \ (*).$$

On y substitue des équations des formes polaires

$$\mathcal{B}_i(x+z) - \mathcal{B}_i(x) - \mathcal{B}_i(z), \ i = 1..n$$

et respectivement

$$\mathcal{A}_i(a+c) - \mathcal{A}_i(a) - \mathcal{A}_i(c), \ i = 1..n$$

On peut alors résoudre des relations (*) par la réduction de Gauss, ce qui donne toutes les différences z et c qui satisfont les relations (*) appliquées aux formes polaires.

Les sommes des tous les z et c trouvés se correspondront, i.e.

$$S(\sum z_i) = \sum c_i.$$

(Remarque : même si un nombre exponentiel de solutions z peut être trouvé, la réduction de Gauss permettra de les exprimer en fonction d'un certain nombre de variables libres z_i et pourra quand même calculer leur somme en temps polynomial par le calcul formel.)

11.7.2 Exemple :

On se place dans $\mathbf{F}_2^5$.
Nous avons choisi au hasard une forme quadratique :

$$\begin{cases} y_0 = 1 + x_1x_2 + x_1x_3 + {x_0}^2 + x_0x_1 + x_4 + {x_2}^2 + x_0x_4 + x_1 \\ y_1 = 1 + x_1 + x_3 + x_4 + x_4x_2 + {x_0}^2 + {x_2}^2 + {x_3}^2 + {x_4}^2 + x_0x_2 + x_0x_1 + x_0x_4 + x_3x_0 \\ y_2 = 1 + x_0x_2 + x_1x_2 + x_4 + {x_3}^2 + x_3x_0 + x_3x_2 + {x_1}^2 + x_1 + x_1x_4 \\ y_3 = 1 + x_4x_2 + {x_4}^2 + x_4 + x_3 + {x_3}^2 + x_3x_0 + x_0 + x_2 + x_4x_3 + {x_0}^2 + x_0x_1 \\ y_4 = x_1 + x_2 + x_4x_2 + {x_1}^2 + {x_4}^2 + x_1x_2 + x_4x_3 + x_0x_4 + x_3x_0 + x_1x_4 \end{cases}$$

et une différence z :

$$z_0 = 0, z_1 = 1, z_2 = 0, z_3 = 0, z_4 = 1.$$

Cela donne des équations de la forme polaire suivantes :

$$
\begin{array}{l}
y_0(x,z) = x_2 + x_3 \\
y_1(x,z) = x_2 \\
dy_2(x,z) = x_1 + x_2 + x_4 \\
y_3(x,z) = x_0 + x_2 + x_3 \\
y_4(x,z) = x_0 + x_1 + x_4 + x_3
\end{array}
$$

Et ainsi on trouve une (unique dans ce cas) relation entre ces équations (numérotées à partir de 0) :

$$y_2 + y_3 + y_4 = 0.$$

□

Pour appliquer le procédé dans l'autre sens on réécrit les équations correspondantes à la forme polaire

$$\mathcal{B}_i(x+z) - \mathcal{B}_i(x) - \mathcal{B}_i(z),\ i = 1..n$$

dans leur totalité :

$$
\left\{
\begin{array}{l}
y_0(x,z) = (z_3 + z_2 + z_0)\, x_1 + z_1 x_3 + z_1 x_2 + z_0 x_4 + (z_4 + z_1)\, x_0 \\
y_1(x,z) = z_0 x_1 + x_3 z_0 + (z_4 + z_0)\, x_2 + (z_2 + z_0)\, x_4 + (z_2 + z_4 + z_1 + z_3)\, x_0 \\
y_2(x,z) = (z_2 + z_4)\, x_1 + (z_2 + z_0)\, x_3 + (z_0 + z_1 + z_3)\, x_2 + z_1 x_4 + (z_3 + z_2)\, x_0 \\
y_3(x,z) = z_0 x_1 + (z_4 + z_0)\, x_3 + z_4 x_2 + (z_3 + z_2)\, x_4 + (z_1 + z_3)\, x_0 \\
y_4(x,z) = (z_2 + z_4) x_1 + (z_4 + z_0) x_3 + (z_4 + z_1) x_2 + (z_3 + z_2 + z_1 + z_0) x_4 + (z_4 + z_3) x_0
\end{array}
\right.
$$

Et maintenant on cherche à résoudre (en z) la relation suivante :

$$y_2 + y_3 + y_4 = 0$$

où l'on substitue les équations écrites plus haut. Cela amène à résoudre :

$$0 = (z_0)\, x_1 + (z_2 + z_0)\, x_3 + (z_0 + z_3)\, x_2 + z_0 x_4 + (z_1 + z_2 + z_3 + z_4)\, x_0$$

Cette équation doit être vraie pour tout x. On a eu de la chance : il y a une seule solution non nulle z, la même que nous avons pris au début :

$$z_0 = 0, z_1 = 1, z_2 = 0, z_3 = 0, z_4 = 1$$

Quelque tests de ce procédé que nous avons effectué (uniquement pour $n = 5$) ont montré que la probabilité d'obtenir une unique solution est assez faible, de l'ordre de 0.1.
Évidemment, comme toujours, il est possible d'obtenir de l'information si'il y a d'autres solutions : par la sommation. Des fonctions ayant beaucoup de symétries risquent toutefois de donner la somme 0. Dans ce cas il y peu de chances que la somme des $\mathcal{A}(z)$ soit nulle aussi.

11.7.3 Application.

Nous allons utiliser précisément cette propriété d'avoir plusieurs solutions pour proposer une variante de l'attaque "va-et-vient dans le milieu".

On procède comme dans l'exemple plus haut, on commence avec $z = d^{(i,0)}$, on trouve l'équation correspondante (on jette les cas où elle n'est pas unique), on récupère toutes les solutions z (y compris celle qu'on a pris au début). On pose : $K(z) =$ somme de toutes les valeurs z obtenues.

Évidemment, K ne fonctionnera pas pour un C^*. Le C^* est un cas difficile pour des nombreuses attaques. Afin de voir ce qui se passe nous avons essayé. Sur le peu d'exemples étudiés, on a constaté qu'il y a unicité de la solution qui empêche d'obtenir une nouvelle l'information.

11.7.4 Les relations conjuguées dans le cas de C^*.

Par contre les relations conjugués, en elles mêmes, fonctionnent parfaitement pour un C^*. Par exemple prenons l'exemple de 11.4. :

$$\mathcal{B} : \begin{cases} y_0 = x_0 + x_2 + x_3 + x_4 + x_0x_1 + x_0x_2 + x_0x_4 + x_1x_3 + x_2x_4 + x_3x_4 \\ y_1 = x_0x_3 + x_0x_4 + x_1x_2 + x_1x_3 + x_1x_4 + x_3x_4 \\ y_2 = x_0 + x_2 + x_3 + x_0x_1 + x_0x_4 + x_1x_2 + x_2x_3 \\ y_3 = x_2 + x_3 + x_0x_1 + x_0x_4 + x_2x_3 + x_2x_4 + x_3x_4 \\ y_4 = x_1 + x_3 + x_4 + x_0x_2 + x_0x_3 + x_1x_2 + x_1x_3 + x_1x_4 \end{cases}$$

$$\mathcal{A} : \begin{cases} b_0 = a_0 + a_0a_4 + a_1a_2 + a_1a_3 + a_2a_3 \\ b_1 = a_2 + a_3 + a_4 + a_0a_1 + a_0a_3 + a_3a_4 \\ b_2 = a_4 + a_0a_1 + a_0a_2 + a_0a_4 + a_1a_2 + a_2a_4 + a_3a_4 \\ b_3 = a_1 + a_2 + a_3 + a_4 + a_0a_4 + a_2a_3 + a_2a_4 \\ b_4 = a_3 + a_0a_2 + a_0a_4 + a_1a_2 + a_1a_3 + a_1a_4 + a_2a_3 + a_2a_4 + a_3a_4 \end{cases}$$

On va prendre $z = (1, 0, 0, 0, 0)$ et $c = (1, 0, 0, 0, 0)$. On calcule facilement :

$$\mathcal{B}(x+z)-\mathcal{B}(x): \quad \begin{cases} x_1+x_2+x_4+1 \\ x_4 \\ x_1+x_4+1 \\ x_1+x_4 \\ x_3+x_2 \end{cases}$$

$$\mathcal{A}(a+c)-\mathcal{A}(a): \quad \begin{cases} a_4+1 \\ a_3+a_1 \\ a_1+a_2+a_4 \\ a_4 \\ a_2+a_4 \end{cases}$$

Ainsi, si l'on néglige les termes constants des équations, on obtient respectivement deux relations suivantes :

$$y_2+y_3=0$$

$$b_0+b_3=0$$

Il est facile à vérifier que ces deux équations se correspondent par T qui avait été trouvé dans 11.4 : Si $y=T(b)$ alors

$$y_2+y_3=b_0+b_3=1.$$

Cette possibilité de trouver n bits d'information sur T étant donné n bits d'information sur S donnés par l'équation initiale

$$S((1,0,0,0,0))=(1,0,0,0,0)$$

est très surprenante.

On sait par ailleurs trouver n bits d'information sur T de façon différente : simplement en "transférant" l'équation $S((1,0,0,0,0))=(1,0,0,0,0)$ de "l'autre côté" :

$$\mathcal{A}(1,0,0,0,0)=T(\mathcal{B}(1,0,0,0,0)).$$

Les deux ensembles d'information, les premiers concernant des lignes de la matrice T et le deuxième concernant ses colonnes, sont presque indépendants. En effet il est facile à montrer qu'en tout on obtient $2n+1$ bits d'information sur T. Cela est très surprenant à cause des propriétés de C^* établies dans 11.4. -il existe $n2^n$ solutions S et T, par conséquent n bits d'information ne suffisent pas pour déterminer l'isomorphisme de façon unique. -certains ensembles de n bits d'information nous permettent d'en obtenir $2n-1$, et donc certains ensembles de $2n-1$ bits d'information ne suffisent pas pour déterminer l'isomorphisme de façon unique. -certains ensembles de $2n$ bits permettent de déterminer la solution de façon unique et sous certaines conditions, en temps polynomial.

Chapitre 12

Le problème MP.

Le problème MP (Morphisme des Polynômes) est une généralisation du problème IP où les applications S et T ne sont plus bijectives.

Plus généralement, on peut également supposer que les polynômes sont définis non plus sur un corps fini K, mais sur un anneau non-commutatif R.
On appellera ce deuxième problème **"problème MP non commutatif"**.

L'exemple suivant est un cas concret du problème MP. On considère deux exemples d'équations suivants :

$$\text{(A)} \begin{cases} b_1 = a_1 a'_1 \\ b_2 = a_2 a'_2 \\ b_3 = a_3 a'_3 \\ b_4 = a_4 a'_4 \\ b_5 = a_5 a'_5 \\ b_6 = a_6 a'_6 \\ b_7 = a_7 a'_7 \end{cases}$$

$$\text{(B)} \begin{cases} y_1 = x_1 x'_1 + x_3 x'_2 \\ y_2 = x_2 x'_1 + x_4 x'_2 \\ y_3 = x_1 x'_3 + x_3 x'_4 \\ y_4 = x_2 x'_3 + x_4 x'_4 \end{cases}$$

Le premier ensemble c'est simplement 7 multiplications. Le deuxième correspond à la multiplication de 2 matrices 2x2 :

$$\begin{pmatrix} y_1 & y_3 \\ y_2 & y_4 \end{pmatrix} = \begin{pmatrix} x_1 & x_3 \\ x_2 & x_4 \end{pmatrix} \cdot \begin{pmatrix} x'_1 & x'_3 \\ x'_2 & x'_4 \end{pmatrix}$$

Le problème est de trouver deux transformations linéaires S et T :

$$(a_1, ..., a_7, a'_1, ..., a'_7) = S(x_1, ..., x_4, x'_1, ..., x'_4)$$

$$(y_1, ..., y_4) = T(b_1, ..., b_7)$$

Telles qu'on peut calculer $\mathcal{B}$ comme une composition de $T \circ \mathcal{A} \circ S$.

Trouver S et T donne souvent une méthode intéressante pour calculer $\mathcal{B}$, par exemple comment calculer le produit vectoriel avec seulement 5 multiplications (voir [10]).

Notre couple $\mathcal{A}$ et $\mathcal{B}$ correspond au problème de multiplier deux matrices 2x2 avec seulement 7 multiplications. Ce problème avait été résolu en 1969 par Strassen ([45]). Voilà sa solution :

$$\begin{cases} a_1 = x_3 - x_4 \\ a_2 = x_1 + x_4 \\ a_3 = x_1 - x_2 \\ a_4 = x_1 + x_3 \\ a_5 = x_1 \\ a_6 = x_4 \\ a_7 = x_2 + x_4 \end{cases} \qquad \begin{cases} a'_1 = x'_2 + x'_4 \\ a'_2 = x'_1 + x'_4 \\ a'_3 = x'_1 + x'_3 \\ a'_4 = x'_4 \\ a'_5 = x'_3 - x'_4 \\ a'_6 = x'_2 - x'_1 \\ a'_7 = x'_1 \end{cases} \qquad \text{et} \qquad \begin{cases} y_1 = b_1 + b_2 - b_4 + b_6 \\ y_2 = b_4 + b_5 \\ y_3 = b_6 + b_7 \\ y_4 = b_2 - b_3 + b_5 - b_7 \end{cases}$$

A ce jour personne n'a établi quel était le nombre minimal de multiplications nécessaire pour multiplier deux matrices 3x3. Le meilleure valeur connue est 23 [23].

Nous travaillons actuellement à l'adaptation de nos algorithmes pour IP au cas de MP.

Dans [19] nous montrons que le problème MP est NP-dur, et sous certaines hypothèses raisonnables IP n'est pas NP-dur. MP est donc probablement plus difficile que IP. Toujours reste-t-il que nos algorithmes pour résoudre IP ne sont pas polynomiaux, et donc il n'est pas du tout exclu qu'on puisse les utiliser pour résoudre MP.

Il convient également de remarquer que la hiérarchie polynomiale s'applique mal à la problématique de IP et MP. En effet, on a fait des énormes progrès en IP, en passant de $\mathcal{O}(q^{n^2})$ à $n^{\mathcal{O}(1)}q^{n/2}$. La complexité n'est pas polynomiale mais permet traiter des cas jusqu'à $n = 80$.

Chapitre 13

Conclusions et perspectives

La classe OR, et plus généralement les cryptosystèmes à polynômes multivariables est très intéressante pour la compréhension de la cryptographie à clef publique dans son ensemble, même si un jour on finit par cryptanalyser tous les systèmes étudiés.

Notre compréhension actuelle nous incite à espérer que certaines variantes de HFE pourraient être solides.

Les systèmes de degré 2 semblent bien explorés. Le tableau suivant résume la situation :

cryptosystème	proposé	cassé
C*	1985	1995
HFE	1996	non
D*	1996	1996
D**	1996	non
HM	1985	1997
HM+	1997	non
Scotch	1995	1997
Super Scotch	1997	non
S-box	1996	1996
(S-box)^2	1996	non

Pour les systèmes de degré 4, on peut imaginer beaucoup de schémas nouveaux, construits à partir des éléments de base de degré 2 dont on sait évaluer les paramètres de sécurité.

Il semble que nous avons bien cerné la difficulté réelle du problème IP et on voit difficilement comment on pourrait encore améliorer la complexité des attaques.

Si on arrive à proposer des méthodes analogues pour MP, cela permettrait des progrès très intéressants en algorithmique, notamment pour la multiplication rapide des matrices.

Il faut continuer dans cette voie.

Bibliographie

[1] Philippe BENEZETH, "La carte à Puce-Histoire" ; CP8 Transac, 21 Février 1997.

[2] BRASSARD Gilles : "Cryptologie Contemporaine" ; Masson 1993.

[3] BRASSARD Gilles : "A note on the complexity" ; IEEE Tran. Inform. Theory, Vol. IT-25, 1979, pp. 232-233.

[4] E. Brickell, A. Odlyzko, *Cryptanalysis, A Survey of Recent Results*, p. 506, in "Contemporary Cryptology", IEEE Press, 1992, edited by Gustavus J. Simmons.

[5] COURTOIS Nicolas "The security of HFE", to be published.

[6] COPPERSMITH Don, WINOGRAD Samuel : "Matrix multiplication via arithmetic progressions" ; J. Symbolic Computation (1990), 9, pp. 251-280.

[7] Hans DOBBERTIN, *Almost Perfect Nonlinear Power Functions on* $GF(2^n)$, paper available from the author.

[8] R. LIDL, H. NIEDERREITER : "Finite Fields" ; Encyclopedia of Mathematics and its applications, Volume 20, Cambridge University Press.

[9] I. BLAKE, X. GAO, R. MULLIN, S. VANSTONE and T. YAGHOOBIAN, *"Applications of Finite Fields"*, Kluwer Academic Publishers.

[10] GUSTAFSON John, ALURU Srinivas : "Massively Parallel Searching for Better Algorithms or, How to Do a Cross Product with Five Multiplications" ; Ames Laboratory, Dept. of Energy, ISU, Ames, Iowa. Accessible à http ://www.scl.ameslab.gov/Publications/FiveMultiplications/Five.html.

[11] Isabelle GUERIN-LASSOUS : "Rapport de Stage-Étude et attaque de l'algorithme de Matsumoto-Imai et de ses généralisations", DEA Intelligence Artificielle et Algorithmique, Université de Caen.

[12] T. JAKOBSEN, R. KNUDSEN : "The interpolation attack on block ciphers" ; Fast Software Encryption, E. Biham editor, 4th int. workshop, Haifa, Israel, LNCS, Springer-Verlag 1997.

[13] Neal KOBLITZ : "Algebraic aspects of cryptography" ; Springer-Verlag, ACM3, 1998, Chapter 4 "Hidden Monomial Cryptosystems", pp. 80-102.

[14] PATARIN Jacques : "Cryptanalysis of the Matsumoto and Imai Public Key Scheme of Eurocrypt'88" ; CRYPTO'95, Springer-Verlag, pp. 248-261.

[15] PATARIN Jacques : "Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP) : two new families of Asymmetric Algorithms" ; EUROCRYPT'96, Springer Verlag, pp. 33-48.

[16] PATARIN Jacques : "Asymmetric Cryptography with a Hidden Monomial" ; CRYPTO'96, Springer Verlag, pp. 45-60.

[17] GOUBIN Louis, PATARIN Jacques : "Asymmetric cryptography with S-boxes" ; juin 1997, à paraî tre dans ISICS'97, Springer-Verlag.

[18] GOUBIN Louis, PATARIN Jacques : "Trapdoor one-way permutations and multivariate polynomials" ; juin 1997, à paraî tre dans ISICS'97, Springer- Verlag.

[19] COURTOIS Nicolas, GOUBIN Louis, PATARIN Jacques : "Improved Algorithms for Isomorphism of Polynomials" ; Eurocrypt 1998, Springer-Verlag.

[20] COURTOIS Nicolas, GOUBIN Louis, PATARIN Jacques : "C*-+ and HM - Variations around two schemes of T. Matsumoto and H. Imai" ; Asiacrypt 1998, Springer-Verlag, pp. 35-49.

[21] GOUBIN Louis, KIPNIS Aviad, PATARIN Jacques : "Unbalanced Oil and Vinegar Signature Schemes" ; Eurocrypt 1999, Springer-Verlag.

[22] Jacques Patarin, Louis Goubin, Nicolas Courtois, + papers of Eli Biham, Aviad Kipnis, T. T. Moh, et al. : *Asymmetric Cryptography with Multivariate Polynomials over a Small Finite Field*; known as 'orange script', compilation of different papers with added materials. Available from J.Patarin@frlv.bull.fr.

[23] Julian D. LADERMAN : "A noncommutative algorithm for multiplying 3x3 matrices using 23 multiplications" ; Bulletin of the American Mathematical Society, Vol. 82, Nr. 1, January 1976.

[24] R. LIDL, H. NIEDERREITER : ""

[25] Tsutomu MATSUMOTO, Hideki IMAI :"A class of asymmetric cryptosystems based on polynomials over finite rings" ; 1983 IEEE International Symposium on Information Theory, Abstract of Papers, pp.131-132, September 1983.

[26] Tsutomu MATSUMOTO, Hideki IMAI :"Algebraic Methods for Construction of Asymmetric Cryptosystems" ; AAECC-3, Grenoble, 15-19 juin 1985.

[27] Tsutomu MATSUMOTO, Hideki IMAI : "Public Quadratic Polynomial- tuples for efficient signature-verification and message-encryption", EUROCRYPT'88, Springer-Verlag 1998, pp. 419-453.

[28] Alfred J. MENEZES, Paul C. van OORSHOT, Scott A. VANSTONE : "Handbook of Applied Cryptography" ; CRC Press.

[29] Alfred J. MENEZES, Paul C. van OORSHOT, Scott A. VANSTONE : "Some computational aspects of root finding in $GF(q^m)$", in Symbolic and Algebraic Computation, Lecture Notes in Computer Science, 358 (1989), pp. 259-270.

[30] G. MULLEN, "Permutation Polynomials over Finite Fields", in "Finite Fields, Coding Theory, and Advances in Communications and Computing", Dekker, Volume 141, 1993, pp. 131-152.

[31] GANTMACHER F. .R. : "The theory of matrices" ; Chelsea Publishing Company, Volume one, chapter VII.

[32] Kaisa NYBERG :"Differentially uniform mappings for crypotography" ; Advances in Cryptology Eurocrypt'93, Lofthus, Norway, LNCS 765, pp. 55-64, Springer-Verlag, 1994.

[33] Michael GAREY, David JOHNSON : "Computers and Intractability, a guide to the theory of NP-completeness", Freeman, p. 251.

[34] Paul van OORSCHOT and Scott VANSTONE : "A geometric approach to root finding in $GF(q^m)$", IEEE Trans. Info. Th., 35 (1989), pp. 444-453.

[35] J. von zur GATHEN and Victor SHOUP, "Computing Frobenius maps and factoring polynomials", Proceedings of the 24th Annual ACM Symposium in Theory of Computation, ACM Press, 1992.

[36] Michael ROSING : "Elliptic Curve Cryptography" ; Manning Publications, USA. The book can be ordered at it's web site, which contains also it's preface, index, the chapter 5, etc. with all the C-programs from the book.

[37] SCHNEIER Bruce : "Applied Cryptography" ; Wiley and sons.

[38] Adi SHAMIR : "Efficent signature schemes based on birational permutations" ; CRYPTO 93, Springer-Verlag, pp1-12.

[39] Adi SHAMIR, Aviad KIPNIS : "Cryptanalysis of the Oil and Vinegar Signature Scheme" ; CRYPTO 98, Springer-Verlag.

[40] Adi SHAMIR, Aviad KIPNIS : "Cryptanalysis of the HFE Public Key Cryptosystem" ; submitted to CRYPTO 99. It can be found at http ://www.univ-tln.fr/∼courtois/hfesubreg.ps

[41] Adi SHAMIR, *"An efficient Identification Scheme Based on Permuted Kernels"*, CRYPTO'89, pp. 606-609.

[42] Don Coppersmith, Jacques Stern, Serge Vaudenay : *Attacks on the birational permutation signature schemes*; Crypto 93, Springer-Verlag, pp. 435-443.

[43] Don Coppersmith, Jacques Stern, Serge Vaudenay, *The Security of the Birational Permutation Signature Schemes*, in Journal of Cryptology, 10(3), pp. 207-221, 1997.

[44] STINSON Douglas R. : "Cryptography, theory and practice" ; CRC Press 1995.

[45] STRASSEN V. : "Gaussian elimination is not optimal" ; Numerishe Mathematik 13, 1969, pp. 354-356.

[46] T. MATSUMOTO, H. IMAI, H. HARASHIMA, H. MIYAKAWA : "Asymmetric cryptosystems using obscure representations of enciphering functions" ; 1983 Natl. Conf. Rec. On Inf. Syst., IECE Japan, S8-5, September 1983 (in japanese).

[47] Michel UGON : "Cartes à Puces", périodique "Techniques de l'Ingénieur", Paris.

[48] Nicolas COURTOIS : HFE security, the unofficial HFE cryptosystem web page. http ://hfe.minrank.org