République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur
et de la Recherche Scientifique

Université de Saida - Dr MOULAY Tahar
Faculté : Technologie
département : Informatique

Mémoire de Master
Option : Sécurité Informatique et Cryptographie

T h è m e

Cryptanalyse Algébrique par Canaux Auxiliaires

Présenté par :
-BELLIL Mohamed
-FRIH Osmane

Encadré par :
Dr.D.BENAMARA

Année universitaire 2019-2020

Remerciements

En première instance, nous tiendrons à exprimer nos remerciements les plus sincères à notre encadreur , Monsieur Djillali BENAMARA, pour nous avoir conseillé, encouragé et soutenu durant toute la période. Il a toujours été disponible et présent pour nous accompagner à toutes les étapes. Grâce à son soutien, nous tiendrons également à remercier chaleureusement Messieurs Miloud BENYAHIA et CHAOUKI , nous les remercions pour leurs conseils et leures disponibilités. toutes l'equipe du département Informatique Enseignants et Administratifs

Nous tiendrons également à remercier nos amis et notres collègues de l'Universite de Saida Dr Moulay Tahar pour les bons moments qu'on a passé ensemble.

Pour Bellil ....

Je remercie chaque membre de ma famille, mon frère , YAHIA sans oublier mes amis Omar ALLEL, Mohamed Rahal, pour notre complicité, et pour leurs encouragements.

Mes plus grands remerciements sont réservés à mon père et ma belle mère.

Souad, mon épouse, aura mon dernier témoignage de reconnaissance. Elle m'a toujours étonné par son grand sens de la responsabilité et son dévouement à notre famille.

Elle m'a supporté, aidé et accompagné à chaque instant. Nos enfants, Youcef, Fatima Zohra, Djelloul et Abdelkader , ont aussi su me soutenir en remplissant ma vie de gaieté et de bonheur.

Pour Frih ....

Je tiens à exprimer toute ma reconnaissance à mon professeur Chenine Abdelkader, Je le remercie de m'avoir encadré, orienté, aidé et conseillé durent toutes ma carrière d'étude .

J'adresse mes sincères remerciements à tous les professeurs, intervenants et toutes les personnes qui par leurs paroles, leurs écrits, leurs conseils et leurs critiques ont guidé mes réflexions et ont accepté de me rencontrer et de répondre à mes questions durant mes recherches.

Je remercie mes très chers parents, Ahmed et Khadidja , qui ont toujours été là pour moi. Je remercie mes frères Tarek,Mokhtar et Farid pour leurs encouragements.

Je tiens ayssi à remercie ma femme, pour leur soutien dont elle a fait preuve pendant

toute la durée de cette thèse,

Enfin, je remercie mes amis Tahar Si Youcef, Sofian Dahkal, Abdelkader Si Ahmed qui ont toujours été là pour moi. Leur soutien inconditionnel et leurs encouragements ont été d'une grande aide . QUE Dieu me les garde et les protège !

Table des matières

Résumé

Ce travail étudie l'application des bases de Gr¨obner à la cryptanalyse de chiffrements par blocs. La base de l'application est un algorithme de résolution systèmes d'équations polynomiales via le calcul de base de Gr¨obner. Dans notre cas,les équations polynomiales décrivent le problème de récupération de clé pour les chiffrements par blocs, c'est-à-dire que la solution de ces systèmes correspond à la valeur de la clé secrète. Premièrement, nous démontrons que la technique de base de Gr¨obner peut être réussie utilisé pour casser les chiffrements par blocs, si la structure algébrique de ces chiffrements est relativement simple. Pour le montrer, nous construisons deux familles de chiffrements par blocs qui satisfont à cette condition. Cependant, nos chiffrements ne sont pas anodins, ils ont un bloc et une taille de clé raisonnables ainsi qu'un nombre de tours acceptable.

De plus, en utilisant des paramètres appropriés, nous obtenons une bonne résistance de ces chiffrements contre la cryptanalyse différentielle et linéaire. En même temps, nous concevoir nos chiffrements de manière à ce que le problème de récupération de clé pour chacun d'eux puisse être décrit par un système d'équations polynomiales simples. De plus, les paramètres des chiffrements peuvent être modifiés indépendamment. Cela rend le construit familles adaptées à l'analyse des attaques algébriques. Pour étudier les vulnérables de tels chiffrements contre l'attaque de base de Gr¨obner, nous avons effectué des expériences en utilisant le système d'algèbre informatique Magma. Les résultats de ces expériences sont donné et analysé. De plus, pour un sous-ensemble de ces chiffrements, nous présentons un méthode pour construire des bases de Gr¨obner de dimension zéro w.r.t. un degré inversé ordre des termes lexicographiques sans réduction polynomiale. Cela réduit la problème de récupération de clé au problème de la conversion de base de Gr¨obner. En utilisant limites de complexité connues pour le dernier problème, nous estimons le maximum résistance de ces chiffrements contre les attaques de base de Gr¨obner.

Nous montrons que notre méthode peut également être appliquée au chiffrement par bloc AES.

Dans la thèse, nous décrivons le problème de récupération de clé AES sous la forme d'un base totale de degré Gr¨obner, expliquez comment cette base Gr¨obner peut être obtenue, et étudier la signification cryptanalytique de ce résultat.

Ensuite, nous étudions la semi-régularité de plusieurs représentations polynomiales pour des chiffrements par blocs itérés. Nous démontrons que le construit La base de Gr¨obner pour l'AES est semi-régulière. Ensuite, nous prouvons que le polynôme les systèmes similaires aux équations quadratiques BES ne sont pas

semi-réguliers ainsi que les systèmes AES d'équations quadratiques sur $GF(2)$ ne sont pas semi-réguliers sur $GF(2)$.

Enfin, nous proposons une nouvelle méthode de cryptanalyse à canal latéral - algébrique attaques par collision - et expliquez-le par l'exemple de l'AES. La méthode est basé sur la technique standard d'analyse de puissance, qui est appliquée pour dériver une information supplémentaire issue d'une implémentation d'un cryptosystème.

Dans notre cas, ces informations concernent des collisions internes généralisées entre les S-boîtes du chiffrement par blocs. Cependant, nous utilisons une nouvelle approche pour récupérer la clé secrète à partir des informations obtenues. Prendre en compte une structure spécifique de l'algorithme cryptographique attaqué, nous exprimons le a détecté des collisions comme un système d'équations polynomiales et utilise Gr¨obner bases pour résoudre ce système. Cette approche offre des avantages significatifs à la fois en termes de mesures et de complexité post-traitement. Comme nous utiliser les non-collisions pour optimiser notre méthode. Pour le chiffrement par bloc AES, nous démontrent plusieurs attaques de collision algébriques efficaces. Le premier d'entre eux fonctionne dans le scénario en clair et nécessite 5 mesures pour dériver la clé secrète complète en quelques heures sur un PC avec une probabilité de succès de 0,93.

Cette attaque avec 4 mesures récupère la clé dans environ 40% des cas. la deuxième attaque fonctionne dans le scénario de paire clair / chiffré connu mais mène pour des résultats plus efficaces : la clé peut être obtenue en quelques secondes hors ligne calculs avec une probabilité de succès de 0,82 pour 4 mesures, et avec probabilité proche de 1 pour 5 mesures. Nous proposons également une attaque de collision algébrique sur l'AES avec 3 mesures. L'attaque a une probabilité de 0,42 et nécessite 4,24 heures de PC après le traitement.

Introduction

L'utilisation de la cryptographie est un droit pleinement reconnu depuis 2004. Cette liberté, finalement assez récente, est une garantie à la liberté de communication et du respect de la vie privée auquel nous sommes tous très attachés. Pour autant, nous ne ressentons pas la nécessité de chiffrer toutes nos lettres et données personnelles afin de les protéger d'éventuels regards indiscrets. L'évocation de la cryptographie provoque même souvent, par méconnaissance de son importance, une réaction d'indifférence sous prétexte qu'un "honnête citoyen n'a rien à cacher". En réalité, depuis que le monde est rentré dans l'âge de l'information, nous avons tous quotidiennement recours à la cryptographie, la plupart du temps probablement sans en être conscient. Elle est devenue indispensable par exemple pour effectuer des transactions financières, notamment lorsqu'on utilise sa carte bancaire, mais aussi pour garantir la sécurité de la plupart des cartes à puces, des passeports numériques, du vote électronique, des services sur internet, des produits télévisuels, des consoles de jeux, des téléphones portables, etc. Essentielle aux systèmes d'informations actuels, la cryptographie procure des moyens de protection des données par des algorithmes cryptographiques assurant la confidentialité mais aussi l'authentification et l'intégrité des données. Les algorithmes cryptographiques sont des fonctions mathématiques très difficiles à inverser sans une information particulière, tenue secrète, appelée clef . La sécurité repose donc, en pratique, sur la difficulté à retrouver la clef secrète à partir d'informations publiques. C'est l'objectif de la cryptanalyse d'estimer cette difficulté, et ainsi d'évaluer la sécurité des algorithmes cryptographiques. Dans ce but, de nombreuses méthodes mathématiques ont été proposées, les plus classiques étant la cryptanalyse linéaire et la cryptanalyse différentielle.

Le sujet de ce travail se rattache à la cryptanalyse algébrique qui consiste à modéliser judicieusement une primitive cryptographique sous la forme d'un système d'équations algébriques. Ce système d'équations est construit de manière à établir une correspondance entre les solutions de ce système et les informations secrètes du cryptosystème en question. La sécurité du cryptosystème repose alors sur la complexité de résolution d'un tel système, complexité qu'il s'agit d'estimer de manière théorique, et pratique, avec des outils fournis par les dernières avancées du calcul formel.

Beaucoup de méthodes puissantes de résolution de systèmes polynomiaux ont déjà été proposées : bases de Gröbner (Buchberger, FGLM, F4, F5) les méthodes de type XSL

Ces techniques de résolution ont été particulièrement efficaces contre un grand nombre de schémas multivariés et contre quelques chiffrements par flot , mais elles restent impraticables contre les chiffrements par blocs .

En effet, la taille des systèmes d'équations polynomiales correspondant est tellement importante (des milliers de variables et d'équations de hauts degrés) qu'on

n'est pas encore capable de prédire correctement la complexité de résolution de tels systèmes polynomiaux. D'un autre côté, les cryptosystèmes asymétriques, comme RSA ou (EC)DSA par exemple, se modélisent par des systèmes d'équations très petits (souvent pas plus d'une seule équation) mais pour lesquels la recherche de solutions entières non triviales pose aussi beaucoup de difficultés.

Ainsi, la cryptanalyse algébrique, comme toute la cryptanalyse en générale, étudie la résistance mathématique des algorithmes cryptographiques.

Elle ne porte donc que sur une partie d'un système cryptographique. Or, la sécurité fournie par un appareil implémentant un algorithme cryptographique ne dépend pas uniquement de la robustesse mathématique de cet algorithme, mais bien de la sécurité de l'ensemble du système. Comme disait P. Kocher en 1999 : "A correct implementation of a strong protocol is not necessarily secure" . En effet, en cryptanalyse on suppose implicitement que les informations disponibles pour un attaquant ne sont limitées qu'à certaines données publiques, en général seulement le chiffré, le message clair et en asymétrique, la clef publique. Or, en pratique, d'autres informations peuvent être récupérées par un attaquant à travers d'autres parties moins sécurisées du système, c'est-à-dire à travers ce qu'on appelle un canal auxiliaire. Par exemple, si la clef privée est sauvegardée dans une zone mémoire mal protégée, il peut être plus facile d'attaquer cette zone que d'attaquer mathématiquement l'algorithme. Les fuites physiques sont un autre exemple de canal auxiliaire possible. En effet, des mesures physiques permettent d'obtenir des informations sur les états internes d'un composant lors du chiffrement ou du déchiffrement. Des méthodes d'analyse, généralement statistiques, de ces mesures peuvent permettre de retrouver des informations sur la clef secrète. On peut citer, par exemple, la mesure avec un oscilloscope de la consommation électrique, qui est à la base de nombreuses attaques particulièrement efficaces . Ou bien encore, celles exploitant les mesures du champ magnétique, du temps de calcul, etc.

Chapitre 1

Introduction générale

1.1 Introduction

Les communications ont toujours constitué un aspect important dans l'acquisition de nouvelles connaissances et l'essor de l'humanité. Le besoin d'être en mesure d'envoyer un message de façon sécuritaire est probablement aussi ancien que les communications elles-mêmes.

Ce qui a donné naissance à la science que nous appelons cryptologie qui a pour objectifs d'assurer l'intégrité, l'authenticité et la confidentialité.

La cryptologie évolue de plus en plus en parallèle avec les tentatives de casser l'un de ses objectifs par un tiers malveillant, afin de subir des modifications du message chiffré transmis ou d'en prendre connaissance.

Actuellement, il y a de plus en plus d'informations qui doivent rester secrètes ou confidentielles telles que les informations échangées par les banques où un mot de passe ne doit pas être divulgué et personne ne doit pouvoir le déduire, c'est pourquoi ce genre d'information est crypté.

Dans ce chapitre, nous allons présenter des généralités sur la cryptographie et ses deux grandes catégories : les chiffrements symétriques et les chiffrements asymétriques qui sont nécessaires pour comprendre le fonctionnement d'un crypto système.[3]

1.2 Quelques définitions

1.2.1 Cryptologie

Depuis toujours, l'être humain a essayé de mettre en place des protocoles servant à un échange sûr de l'information tels que les cachets d'enveloppes et les signatures manuscrites.

De nos jours ces mêmes procédés trouvent leurs analogues dans le monde informatique et ceci grâce à la cryptologie.

La cryptologie est la science qui traite la façon de modifier ou de masquer une information afin de la rendre incompréhensible aux yeux des autres ; à l'exception de celui à qui est destinée. Celui-ci lui rendra son aspect initial grâce au secret qu'il détient qui est la clé.

La cryptologie est basée sur les mathématiques et l'informatique et comporte deux branches : la cryptographie et la cryptanalyse.[3]

1.2.2 Cryptographie

La cryptographie concerne la transformation d'un message (texte, image, chiffres) intelligible vers un message codé, incompréhensible à tous sauf pour les détenteurs de la clé de chiffrement.

C'est une discipline qui étudie les méthodes pour assurer le secret et l'authenticité des messages. Le terme « cryptographie » vient du grec « kriptos » (caché) et « graphein » (écrite).

L'art de définir des codes est la cryptographie (un spécialiste de cryptographie est un cryptographe). Elle définit les moyens et les méthodes utilisés pour assurer la confidentialité (et d'autres fonctions de sécurité) des informations que l'on va acquérir, stocker, traiter, diffuser et échanger.[3]

1.2.3 Cryptanalyse

Depuis l'existence des codes secrets, on a cherché à les casser et à comprendre les messages chiffrés bien que l'on n'en soit pas le destinataire légitime, autrement dit les décrypter. Décrypter ou casser le code c'est parvenir au texte en clair sans posséder au départ les règles ou documents nécessaires au chiffrement. L'art de casser des codes est la cryptanalyse (un spécialiste de cryptanalyse est un cryptanalyste, cryptologue ou casseur de codes).[3]

1.2.4 Objectifs de la Cryptographie

Les principaux services offerts par la cryptographie moderne sont :

- Confidentialité : assurer que les données concernées ne pourront être dévoilées que par les personnes autorisées.

- Intégrité : assurer que les données ne seront pas altérées pendant leur transmission ou leur stockage.

- Authentification/Identification : prouver l'origine d'une donnée ou s'assurer de l'identité d'une personne.

- Non-répudiation : garantir que les actions ne seront pas reniées.[3]

1.2.5 Catégories de chiffrements

Les techniques cryptographiques se découpent en deux grandes parties :

- La cryptographie à clés secrètes ou cryptographie symétrique.
- La cryptographie à clés publiques ou cryptographie asymétrique.

Ils ont tous deux leurs avantages et leurs inconvénients. La différence qui existe entre ces deux types se situe au niveau de la clé.[3]

Cryptographie symétrique

La cryptographie symétrique est basée sur l'utilisation d'une clé secrète identique pour le chiffrement et le déchiffrement. Le schéma de principe de la cryptographie à clé secrète est représenté sur la figure 1.1.

Figure 1.1. Schéma de principe de la cryptographie symétrique

La cryptographie symétrique est très utilisée et se caractérise par une grande rapidité et des opérations simples : décalage de certains bits, XOR bit à bit, permutations de certains bits, etc., et par des implémentations aussi bien software que hardware ce qui accélère nettement les débits et autorise son utilisation massive.

De plus, le message (clair) est séparé en blocs successifs, chaque bloc étant chiffré individuellement. Cela permet de disposer de systèmes de chiffrement très performants et très simples.

Les deux parties communicantes doivent se mettre d'accord sur la clé secrète utilisée. Le problème est qu'on ne dispose pas d'un canal de communication sûr pour échanger les clés.

Cryptographie asymétrique

La cryptographie à clé publique s'attache à résoudre le problème de gestion des clés. Son principe fondamental est de donner à chaque utilisateur deux clés associées, l'une secrète et l'autre rendue publique.

Afin de chiffrer un message à l'intention d'un utilisateur, l'idée consiste à n'utiliser que sa clé publique alors que le déchiffrement doit nécessiter la connaissance de la clé secrète. Ce concept naturel permet de communiquer de manière confidentielle sans avoir à partager la moindre information secrète initialement. Le schéma de principe de la cryptographie à clé publique est représenté sur la Figure 1.2

Figure 1.2. Schéma de principe de la cryptographie asymétrique

Du fait de l'utilisation de grands nombres premiers, les crypto systèmes asymétriques nécessitent une quantité de calcul importante, ce qui les rend très lents par rapport aux systèmes symétriques.

Le RSA, nommé d'après les noms de ses inventeurs (Rivest, Shamir et Adelman), est le premier algorithme asymétrique publié en 1977. La sécurité du RSA vient de la difficulté de factoriser des grands nombres premiers : s'il est facile de multiplier deux grands nombres premiers, il est très difficile de décomposer le très grand nombre obtenu en ses deux facteurs premiers quand on ne les connaît pas.

Les clés publique et privée sont une fonction d'un couple de grands nombres (1024 bits ou plus) premiers. Découvrir le texte en clair à partir de la clé publique et du texte crypté est conjecturé comme équivalent à factoriser le produit des deux grands premiers.

La clé publique contient le produit de deux nombres premiers très grands, et un autre nombre qui lui est propre. L'algorithme de chiffrement utilise ces nombres pour chiffrer le message par blocs. L'algorithme de déchiffrement nécessite quant à lui l'utilisation d'un nombre contenu uniquement dans la clé privée.

Les clés RSA sont habituellement de longueur comprise entre 1024 et 2048 bits.

Comparaison entre les chiffrements symétrique et asymétrique

Les chiffrements symétriques et asymétriques présentent chacun des avantages et des inconvénients.

La principale difficulté des algorithmes de chiffrement symétrique ou à clés secrètes réside dans la sécurité de l'échange des clés. Un autre inconvénient est que tout couple d'utilisateurs doit au préalable s'entendre sur une clé commune. La gestion des clés devient vite problématique. Toutefois, les systèmes symétriques sont très efficaces. Ils demeurent sûrs, rapides et peuvent chiffrer et déchiffrer une grande quantité de données en des temps records.

Les cryptosystèmes asymétriques actuels sont beaucoup plus lents que leurs homologues symétriques, et nécessitent de plus longues clés. Leur avantage principal réside dans la simplicité de la gestion des clés : le secret n'est pas partagé, les clés publiques peuvent être publiées dans l'annuaire, et le nombre de clés est bien inférieur lorsque plusieurs personnes veulent communiquer entre elles de façon confidentielle.

Dans ce chapitre, nous avons présenté une introduction à la cryptographie où les deux catégories de cryptographie symétrique et asymétrique ont été exposés.

La cryptographie symétrique est rapide car elle utilise des clés de petites tailles et facile à implémenter sur matériel car elle utilise des opérations simples telles que des décalages, des XOR et des permutations et c'est à elle que nous nous intéressons.

Le prochain chapitre décrira le système cryptographique symétrique AES qui est la cible de notre implémentation logicielle.

1.3 Types des cryptanalyses :

1.3.1 Cryptanalyse linéaire

• C'est une attaque à texte clair connu contre les protocoles de cryptographie dont la confusion est faible.

• Texte clair connu. L'attaquant dispose de un ou plusieurs message(s) clair(s) avec le(s) message(s) crypté(s) correspondant, tous cryptés avec la même clé. L'attaquant cherche à retrouver (de l'information sur) la clé.

• Une idée. Trouver des relations linéaires de dépendance de probabilités exceptionnelles entre les bits d'entrée et de sortie.

En effet, une relation linéaire ne peut pas être vraie pour tous les messages sinon le protocole a une faiblesse.

• Idée générale proche de la cryptanalyse différentielle (attaque à clairs choisis)

• On utilise des approximations linéaires des algorithmes de chiffrement par bloc

• La cryptanalyse linéaire consiste à simplifier l'algorithme de chiffrement en faisant une approximation linéaire.

• En augmentant le nombre de couples disponibles, on améliore la précision de l'approximation et on peut en extraire la clé.

• La cryptanalyse linéaire s'intéresse aux relations linéaires entre les bits au cours de l'algorithme [14]

Forme linéaire • Soit $F : \{0,1\}^n \to \{0,1\}^n$

• Une forme linéaire $\lambda : \{0,1\}^n \to \{0,1\}^n$ est définie par un masque $a = (a_1 \dots a_n) \cdot \lambda(x_1 \dots x_n) = a_1.x_1 \oplus \dots \oplus a_n.x_n$

Caractéristique linéaire • Une caractéristique linéaire de F est un couple de formes linéaires (λ_1, λ_2) ayant pour masques associés a_1 et a_2 telles que $\lambda_1(x) = \lambda_2 F(x)$

• avec probabilité p (prise sur tous les x possibles).

Méthodologie On suppose qu'il existe une F_2-combinaison linéaire des bits d'entrée et de

sortie qui ait lieu avec une probabilité nettement supérieure ou nettement

inférieure à $1/2$. Autrement dit qu'il existe $(i_1, i_2 \dots i_n)$ et$(j_1, j_2 \dots j_n)$ tels que si $x = (x_1 \dots x_n)$ sont les bits d'entrée (du message en clair) considérés comme des variables aléatoires définies sur $\{0,1\}$et $(y_1 \dots y_n)$ sont les bits de la sortie (du message chiffré) considérés comme des variables aléatoires définies sur $\{0,1\}$ on ait $Pr\{X_{i_1} \oplus X_{i_2} \oplus \cdots \oplus X_{i_n} \oplus Y_{i_1} \oplus Y_{i_2} \oplus \cdots \oplus Y_{i_n} = 0\} >> 1/2$ Le principe est alors d'approximer une partie de l'algorithme de chiffrement par cette combinaison linéaire sur F_2.

1.3.2 Cryptanalyse différentielle

• Il s'agit d'une attaque à clairs choisis.

• La cryptanalyse différentielle s'intéresse à l'évolution les différences $x_i + x_i^{'}$ pour deux clairs x_i , $x_i^{'}$. On détermine que,

si $x_i + x_i^{'} = \alpha$, alors $x_{r-1} + x_{r-1}^{'} = \beta$ avec une probabilité non négligeable.

• On utilise cela pour déterminer la clé inconnue k_r à partir de plusieurs messages x et de leurs chiffrés x_r obtenus par $\Bbbk_r$

Le principe général de cette attaque consiste à considérer des couples de clairs X et $X^{'}$ présentant une différence ΔX fixée et à étudier la propagation de cette différence initiale à travers le chiffrement.

On traite les couples d'entrée et de sortie comme des variables aléatoires que l'on note $X, Y, \Delta X, \Delta Y$.

Les différences sont définies par une loi de groupe, en général le xor bit à bit.

Cette attaque utilise la faiblesse potentielle de la fonction itérée f dans une dérivation à l'ordre 1.

La cryptanalyse différentielle utilise la comparaison du XOR de deux entrée avec le XOR des deux sorties correspondantes. On considère

$x' = (x_1^{'}, x_2^{'}, ..., x_n^{'})$ et $x^{''} = (x_1^{''}, x_2^{''}, ..., x_n^{''})$ deux entrées et $y' = (y_1^{'}, y_2^{'}, ..., y_n^{'})$ et $y^{''} = (y_1^{''}, y_2^{''}, ..., y_n^{''})$les sorties correspondantes.

• On note $\Delta x = x\prime \oplus x^{''}$, $\Delta y = y\prime \oplus y$

• Si le système cryptographique était parfait alors la probabilité pour qu'un Δy provienne d'un Δx devrait être de $1/2^n$ où n est le nombre de bits de$X$.

La cryptanalyse différentielle exploite le fait qu'il peut arriver qu'un Δy particulier arrive avec une très grande probabilité , $p >> 1/2^n$, d'un Δx particulier.

Le couple $(\Delta x, \Delta y)$est appelée une différentielle.

On suppose que le cryptanalyste dispose d'un grand nombre de quadruplets $(x\prime, x", y\prime, y")$ où la valeur de Δx est fixée et que tous les textes sont chiffrés

avec la même clef inconnue K.

• Pour chacun des quadruplets on commence par déchiffrer $y\prime$et $y"$ en utilisant toutes les sous clefs candidates pour le dernier étage.

• On commence par regarder les caractéristiques différentielles des S-boites. On remarque que dans ce cas $\Delta y = S(x\prime) \oplus S(x^{''})$. [14]

1.3.3 Cryptanalyse Algébriques [1]

Les attaques algébriques sont des attaques à clair connu qui exploitent des relations algébriques entre les bits du clair, ceux du chiffré et ceux de la clef secrète. La connaissance de plusieurs couples clairs-chiffrés fournit un système d'équations dont les inconnues sont les bits de la clef secrète. Ces derniers peuvent alors être retrouvés en résolvant le système, ce qui est possible s'il est de degré faible, de petite taille ou qu'il possède une structure particulière.

Ces attaques sont utilisées contre, par exemple :

– Stream Ciphers (algorithme de chiffrement à flot) E_0 (Bluetooth, 2001, Armknecht et al., 2008), TOYOCRYPT (Courtois et al., 2003), LILI 128 (Courtois et al., 2003)

– Block Ciphers (algorithme de chiffrement par bloc) A.E.S. (Courtois et al., 2002), Serpent (Courtois et al., 2002)

Chiffrement à flot

Une des raisons pour lesquelles il ne faut pas choisir comme fonction de filtrage une fonction de petit degré est qu'un tel choix rendrait le système vulnérable à une

attaque algébrique de base. En effet, si la fonction est de petit degré d, chaque bit de la suite chiffrante st, s'écrit comme une fonction de degré d en les l bits de l'état initial, puisque l'état du registre à l'instant t est une fonction linéaire de son état initial.

L'exemple du LFSR On peut exprimer s_t comme une fonction de degré 2 en

$(u_0, ..., u_4)$ qui sont les 5 bits d'initialisation du registre : $s_0 = u_3u_4 + u_0u_1 + u_4$.

En utilisant le fait que la suite $(u_t)_{t\geq 5}$ produite par le L.F.S.R. vérifie la récurrence $u_t = u_t - 2 + u_t - 5$ on en déduit qu'à l'instant $t = 1$, on a :

$$s_1 = u_4u_5 + u_1u_2 + u_5.s_1 = u_0u_4 + u_3u_4 + u_1u_2 + u_3 + u_0.$$

Puisque $u_5 = u_3 + u_0$. Au temps $t = 2$, on a :

$$s_2 = u_0u_1 + u_1u_2 + u_3u_4 + u_1u_3 + u_2u_3 + u_1 + u_4.$$

La connaissance de N bits de suite chiffrante permet donc d'écrire un système de N équations de degré 2 à 5 variables. Un tel système peut se résoudre grâce à des algorithmes de résolution de systèmes algébriques tels que les algorithmes de bases de Gröbner. Une méthode moins efficace mais plus simple consiste à assimiler tous les monômes de degré inférieur ou égal au degré des équations à des nouvelles variables,il s'agit de la linéarisation.

Dans l'exemple, on pose donc $x_0 = u_0, ..., x_4 = u_4, x_5 = u_0u_1, x_6 = u_0u_2, ..., x_{14} = u_3u_4$.

Chaque équation de degré 2 s'écrit donc comme

une équation linéaire en $x_0, ..., x_{14}$, par exemple :

$s_2 = x_6 + x_9 + x_{14} + x_{10} + x_{12} + x_1 + x_4$.

La donnée de 15 équations de cette forme fournit donc un système linéaire de 15 équations à 15 inconnues que l'on peut résoudre par une simple élimination de Gauss.

La complexité de l'algorithme est donc de l'ordre de :

$$\left[\sum_{i=1}^{d} \binom{l}{i}\right]^3$$

où d est le degré de la fonction de filtrage f et l la longueur du *L.F.S.R.* Ce nombre d'opérations n'est donc plus accessible dès que le degré de la fonction est élevé, quand on considère des registres de longueur cryptographique, c'est-à-dire quand l dépasse 100

Chiffrement par blocs Dans (Raddum et al., 2007), H. Raddum et I. Semaev proposent une nouvelle attaque algébrique. L'idée générale de cette attaque est de mettre en équation toutes

les solutions possibles et de retirer au fur et à mesure les solutions qui ne peuvent correspondre. Au final, on se retrouve alors avec toutes les clés qui nous permettent

de passer du clair au chiffré. Si on ne trouve pas l'unicité de la solution on peut par exemple ajouter au système un autre couple de clair/chiffré. La nouveauté de cette attaque est introduite dès la mise en équation.

Mise en équation Pour les cryptosystèmes symétriques ce sont les fonctions de chiffrement non linéaires qui font monter le degré des équations, et qui rendent donc difficile une attaque

algébrique. Par exemple, pour les cryptosystèmes Data Encryption Standard (D.E.S.), Advanced Encryption Standard (A.E.S.) et PRESENT, ce sont les Sbox qui jouent le rôle de fonction non linéaire.

Les auteurs proposent de poser pour chaque partie non linéaire un système linéaire qui est constitué d'une matrice système et d'une matrice solution (Raddum et al.,2006).

Pour cela, on considère les variables dans F_2 qui représentent tous les bits issus des fonctions non linéaires et de la clé. Dans la figure 1, les "X" représentent tous les octets qui sont mis en équations.

De la manière que les variables sont définies, chaque variable peut être exprimée en combinaison linéaire. Ces combinaisons linéaires dépendent des variables, des bits du clair et des bits du chiffré. Pour construire la matrice système on exprime pour les premières lignes les relations linéaires qui lient les bits des variables d'entrée de

la fonction non linéaire aux autres variables. Pour les dernières lignes de la matrice système, on fait de même pour les bits de sortie.

Ensuite, on construit la matrice solution du système linéaire en mettant en face des équations d'entrées toutes les entrées possibles et en dessous la sortie correspondante.

Résolution Pour résoudre ce système les auteurs définissent des opérations sur les symboles : la mise en accord, et le collage.

Mise en accord La mise en accord de deux symboles $S_i = (A_i, L_i) et S_j = (A_j, L_j)$ est un procédé qui a pour but de rendre consistant le système concaténé.

Définition 1. *On dit que le symbole S_i est en accord avec S_j si $\forall a_i \in L_i, \exists a_j \in L_j$*

tel que

$$\binom{A_i}{A_j} X = \binom{a_i}{a_j} \qquad \text{soit consistant.}$$

On a montré dans cet partie le rôle essentiel des attaques algébriques dans la cryptanalyse actuelle. Tant en chiffrement à flot qu'en chiffrement par blocs, les attaques algébriques contribuent fortement à affaiblir la sécurité de ces systèmes. De récents travaux apportent régulièrement des améliorations dans ce sens. On a brièvement présentés ce type d'attaque sur l'exemple du LFSR et E0 ; et détaillé la mise en place de ces attaques sur les algorithmes de chiffrement par blocs avec la construction de H.Raddum et I. Semaev. [1]

1.4 Attaques par canaux auxiliaires [11]

1.4.1 Introduction

On dénomme attaque par canaux auxiliaires (ou aussi attaque physique ou Side Channel Attack) toute attaque qui exploite l'implémentation d'un algorithme cryptographique sur un support physique. En effet, la sécurité fournie par un appareil mettant en œuvre un algorithme cryptographique ne dépend pas uniquement de la robustesse mathématique de cet algorithme, mais bien de la sécurité de l'ensemble du système.

En pratique, des informations sur le fonctionnement interne du système peuvent fuir pendant l'exécution de l'algorithme. Un attaquant peut ainsi récupérer les fuites d'information dépendant des opérations effectuées et des données manipulées, en particulier la clef secrète ou des variables intermédiaires dépendant de cette clef. On dit alors que l'attaquant a utilisé un canal auxiliaire pour collecter des fuites d'information. L'analyse de ces données permet ensuite de mener des attaques particulièrement efficaces.

Un grand nombre de techniques ont été proposées pour exploiter des canaux auxiliaires. On peut les distinguer en deux grandes catégories : les attaques passives et les attaques actives.

Les attaques passives sont celles qui se contentent d'observer un appareil cryptographique sans modifier son exécution et récupèrent généralement des informations par les variations de ses propriétés physiques, comme par exemple, le temps d'exécution, le rayonnement électromagnétique, ou la consommation électrique. L'interprétation de ces variations nécessite souvent le choix d'un modèle de fuite, c'est-à-dire d'une fonction de référence explicitant le lien entre les données manipulées et ces variations.

Dans les attaques actives, l'attaquant tente de perturber l'exécution afin de provoquer un comportement anormal de l'appareil. Par exemple, en injectant une faute durant un calcul pour modifier le résultat attendu à la sortie de l'algorithme, et récupérer ainsi des fuites d'information exploitables.

L'infection par des logiciels malveillants, tels que les trojans, peut aussi être considérée comme des attaques actives.

1.4.2 Attaques passives

La première publication d'une attaque par canaux auxiliaires contre une implémentation cryptographique a été proposée par Kocher en 1996.

Cette attaque tire profit du temps d'exécution de certaines opérations qui dépendent de la clef secrète. Les attaques par canaux auxiliaires ont ensuite été

largement étudiées, en particulier celles exploitant la consommation du courant électrique durant l'exécution d'un algorithme cryptographique.

Un circuit électronique consomme une certaine quantité de courant pour chaque opération qu'il effectue. Pour les circuits numériques en particulier, ce courant permet d'alimenter les portes logiques, de représenter les données dans les mémoires et de faire transiter ces données sur les bus de transmission.

Par exemple, pour les circuits CMOS (principale technologie utilisée dans les dispositifs électroniques), la consommation totale de courant est la somme des consommations des cellules logiques le constituant. La consommation totale dépend donc surtout du nombre de cellules dans le circuit, de la façon dont elles sont reliées entre elles et de leurs consommations individuelles. Or, une cellule consomme surtout du courant lors d'un changement d'état. La consommation électrique du composant (ainsi que le champ électromagnétique) dépend donc fortement des données manipulées par le circuit.

D'un point de vue pratique, l'attaque commence par une première étape constituant la campagne d'acquisitions. Elle nécessite quelques équipements pour mesurer le canal auxiliaire pendant l'exécution de l'algorithme. Typiquement, le dispositif cryptographique (généralement un système embarqué) est commandé par un ordinateur à travers une interface dédiée. Pour une attaque visant une carte à puce par exemple, un lecteur de carte commandé par un ordinateur peut être utilisé pour lancer des chiffrements et récupérer les données chiffrées. Un appareil de mesure envoie alors à l'ordinateur les mesures effectuées tout au long du chiffrement pour un traitement ultérieur.

Dans le cas d'une analyse de consommation du courant, on utilise un oscilloscope pour mesurer la différence de potentiel d'une résistance placée entre la source de courant et l'appareil visé. Dans le cas d'une analyse du champ électromagnétique, le signal est fourni par une sonde électromagnétique branchée à un oscilloscope.

Pour chaque exécution de l'algorithme, on obtient ainsi un ensemble de mesures qui forme une courbe en fonction du temps pour un couple (clef, message) donné et qu'on appelle trace de consommation, ou trace d'émission.

Les traces contiennent toujours du bruit, c'est-à-dire qu'une part du signal mesuré est constitué de signaux indésirables variant aléatoirement. Le bruit a différentes sources, par exemple il peut venir de la source de courant, des différents composants électroniques, d'émissions thermiques ou électromagnétiques, ou encore de la qualité des appareils de mesure. La qualité des mesures dépend ainsi de la quantité de bruit qu'elles contiennent par rapport à la composante dépendant uniquement des données manipulées et des opérations effectuées.

Différentes techniques d'analyse des traces permettent ensuite de retrouver les informations secrètes. Lorsque le bruit est important, ces attaques consistent généralement à appliquer des méthodes statistiques sur un grand nombre de traces.

Analyse différentielle de consommation (DPA) L'analyse différentielle de consommation (DPA pour Differential Power Analysis) est la principale attaque par analyse de consommation électrique.

Elle a été proposée pour la première fois par P. Kocher, J. Jaffe et B. Jun en 1999 ([KJJ99]). Contrairement à la SPA, la DPA permet d'exploiter des traces de consommation très bruitées. En contrepartie, la DPA nécessite souvent un grand nombre de traces pour l'utilisation d'outils statistiques.

Toutes les traces doivent cependant provenir de chiffrements effectués avec une clef secrète fixée.

En pratique, l'attaque commence par une campagne d'acquisition des traces de consommation de l'exécution de l'algorithme sur des données différentes.

Suivant le contexte de l'attaque, chaque trace est généralement associée à un chiffré seul ou un texte clair. Une étape intermédiaire de l'algorithme doit ensuite être choisie telle que chaque bit b_i de cette étape intermédiaire puisse s'exprimer comme une fonction $b_i = f(d, k_i)$ du texte connu d et de k_i un petit nombre de bits de la clef. Par exemple, lors d'une attaque contre une implémentation d'un chiffrement par blocs avec clair connu, on peut commencer par cibler un bit de sortie de la première boîte-S du premier tour, dont la valeur ne dépend plus que des premiers bits de la clef.

Il ne reste plus qu'à appliquer une fonction de distinction sur la cible bi choisie afin de déterminer la valeur la plus probable des bits de clef k_i correspondant. La différence de moyenne est la fonction de distinction originairement proposée pour la DPA. A partir d'une hypothèse sur la valeur des quelques bits k_i de la clef, la valeur prévue du bit $b_i = f(d, k_i)$ ciblé est calculée.

Pour chaque hypothèse, les traces sont alors séparées en deux catégories, celles pour lesquelles le bit b est prévu égal à 1 et celle où il est prévu égal à 0. La différence Δk_i des moyennes des deux ensembles est calculée pour toutes les hypothèses sur k_i. Pour l'hypothèse correspondant à la bonne valeur de k_i, on s'attend à observer un pic de consommation à l'instant où les traces dépendent de la valeur du bit ciblé b_i. Pour toutes les autres hypothèses, les moyennes des traces par catégorie tendent à devenir indépendantes de la valeur du bit ciblé b_i, et leurs différences tendront donc à s'annuler lorsque le nombre de traces augmente.

On peut ainsi retrouver toute la clef secrète en recommençant sur un autre bit cible.

1.4.3 Attaques actives

Dans le cadre des attaques par canaux auxiliaires, les attaques actives désignent principalement les injections de fautes. Les attaques par fautes ont été introduites par Boneh, DeMillo et Lipton en 1997 concernant des cryptosystèmes à clef publique. Elles ont ensuite été étendues aux cryptosystèmes à clef secrète par Biham et Shamir.

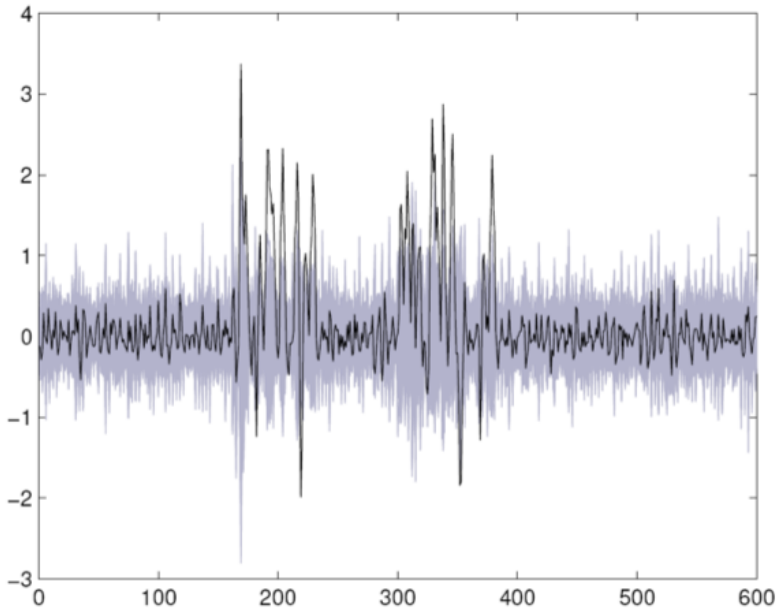


FIGURE *1.1 – Attaque DPA contre une implantation de l'AES. La différence correspondant à l'hypothèse correcte est tracée en noir*

1.5 Cryptanalyse Algébriques par Canaux Auxiliaires [11]

En 2002, des attaques algébriques utilisant des systèmes d'équations surdéfinis ont été proposées comme une cryptanalyse potentiellement très puissante techniquement contre les chiffrements par blocs. Cependant, bien qu'un certain nombre d'expériences convaincantes aient été réalisées contre certains algorithmes réduits, il n'est pas clair si ces attaques peuvent être appliquées avec succès en général et à une grande classe de chiffrements. Dans cet partie, nous montrons que les attaques algébrique peuvent être combinées avec des attaques par canal auxiliaire d'une manière très efficace et naturelle. A titre d'illustration, nous les appliquons au bloc cipher AES qui est une première cible stimulante, en raison de sa simple structure algébrique. Les attaques proposées ont un certain nombre de caractéristiques intéressantes :

(1) elles exploitent les fuites d'informations de tous les chiffrements arrondis,

(2) dans des contextes d'implémentation courants (par exemple en supposant un modèle de fuite de poids de Hamming), ils récupèrent les clés de chiffrement par blocs après l'observation d'un seul cryptage,

(3) ces attaques peuvent réussir le scénario contradictoire en clair inconnu / texte chiffré et

(4) directement contourner les contre-mesures telles que le masquage booléen. Finalement, nous soutenons que les attaques algébriques par canal auxiliaire peuvent tirer parti de tout type de fuite physique, conduisant à un nouveau compromis entre la robustesse et caractère informatif de l'extraction d'informations par canal auxiliaire.

Dans la cryptanalyse classique contre les chiffrements par blocs, un adversaire est généralement fourni avec les entrées / sorties d'un algorithme cible. Attaques secondaires en plus lui fournir des informations partielles sur les valeurs intermédiaires de chiffrement, fuite par un appareil effectuant un calcul cryptographique. Ces attaques sont donc beaucoup moins générales - puisqu'elles sont spécifiques à une implémentation donnée - mais souvent beaucoup plus puissante que la cryptanalyse classique. Par conséquent, ils sont considérés très sérieusement par les fabricants de dispositifs cryptographiques (par exemple, les cartes à puce).

Suite à la publication de la première analyse de puissance différentielle (DPA) dans le fin des années 90 , divers types d'attaques par canal auxiliaire ont été proposés pour effectuer des récupérations de clés efficaces. La plupart de ces techniques partagent une stratégie de division pour vaincre dans laquelle différentes parties de une clé cible (par exemple, des octets physiques, généralement) est récupérée séparément. Ils aussi exploitent généralement les fuites correspondant aux premiers (ou derniers) tours d'un bloc chiffre, où la diffusion est suffisamment faible pour certaines parties de l'intermédiaire calculs dépendants de la clé pour être facilement énumérés et prédits.

En fait, ces attaques par canal auxiliaire sont assez exigeantes en fuite car ils utilisent des mesures physiques pour identifier exactement les octets clés.

En outre, ils n'exploitent généralement pas les faiblesses particulières des chiffrements par blocs.

Par conséquent, une question intéressante est de savoir si un adversaire pourrait utiliser des canaux latéraux pour récupérer des cibles simples plutôt que des valeurs d'octets clés exactes, puis utiliser ces informations partielles dans une étape d'analyse cryptographique hors ligne plus élaborée. Dans en d'autres termes, pouvons-nous arrêter de mesurer plus tôt et avoir encore la complexité d'un récupération de clé qui ne croît pas de façon exponentielle avec la taille de la clé ?

Dans cette partie, nous répondons positivement à cette question et montrons que la combinaison de puissantes attaques de canal auxiliaire (de type modèle) avec une analyse cryptographique algébrique effectuer des récupérations de clés avec un accès extrêmement restreint aux périphériques cibles.

Pourtant, l'attaque est générale et peut fonctionner d'une manière flexible qui inclut les deux phases suivantes. Tout d'abord, l'adversaire sélectionne autant de calculs intermédiaires dans l'algorithme cible que possible et mesure leur fuite physique. Pour chaque de ces calculs intermédiaires, il récupère des informations partielles. Ce l'information partielle peut être représentée par une fonction surjective dont la la sortie a été récupérée grâce à une attaque par canal auxiliaire. À titre d'exemple typique, si un appareil fuit une information qui est fortement corrélée avec le Hamming poids des résultats des calculs intermédiaires cibles, cette fonction pourrait être la fonction de poids de Hamming. Mais tout autre type de fonction (et donc fuite modèle) pourrait être envisagée. C'est finalement le choix de l'adversaire de sélectionner un cible à la fois informative et robuste. Aux extrêmes, une fonction bijective est le plus informatif, mais récupérer sa sortie par une attaque par canal auxiliaire peut nécessitent plusieurs mesures - et une fonction surjective avec un seul possible la valeur de sortie ne donne aucune information. Puis, lors d'une deuxième phase (hors ligne),l'adversaire exploite cette information partielle sur l'algorithme intermédiaire valeurs avec une attaque algébrique. Autrement dit, il écrit le chiffrement par bloc comme un système d'équations quadratiques (ou cubiques,...) et ajoute les fonctions précédemment définies avec des sorties connues vers le système. En pratique, l'approche que nous suivons dans ce papier consiste à convertir le système d'équations représentant le chiffrement par blocs en un Problème SAT et utiliser un solveur automatisé pour effectuer les récupérations de clés. Il s'avère que cette solution a donné de très bons résultats. Cependant, cela reste un question ouverte pour déterminer les meilleures techniques à cet effet.

Les attaques proposées diffèrent de la plupart des attaques par canal auxiliaire connues auparavant dans un certain nombre d'aspects intéressants. Premièrement, ils exploitent potentiellement la fuite de tous les tours de chiffrement (le DPA classique exploite généralement le premier ou le dernier tour seulement). Deuxièmement, ils peuvent réussir dans un adversaire inconnu en clair / texte chiffré contexte (DPA classique nécessite généralement la connaissance des textes en clair ou les textes chiffrés). Troisièmement, ils nécessitent beaucoup moins d'observations pour réussir. Dans certains contextes d'implémentation raisonnables, nous montrons que la trace de fuite d'un seul le chiffrement peut être suffisant pour effectuer une récupération de clé complète. Ceci implique que les constructions basées sur des stratégies de recléage telles que [19, 20] peuvent parfois être rompues dans la pratique. Finalement, ils peuvent traiter des chiffrements de bloc protégés par un masquage des contre-mesures. En particulier, nous montrons des expériences qui cassent ces conceptions masquées avec une seule trace, presque aussi facilement que celles non protégées.

En résumé, les attaques par canal auxiliaire classiques peuvent être considérées comme une combinaison de deux sous-problèmes :

(1) comment récupérer efficacement des informations partielles sur certains parties d'un état chiffré ? "

(2) comment exploiter efficacement cette information partielle ?". Les attaques algébriques par canal auxiliaire soulèvent une nouvelle question, à savoir : qui des informations partielles devons-nous essayer de récupérer ? ". Elle se rapporte à la fois à la précédente mentionné le compromis entre robustesse et informativité et à la sélection des meilleures classes clés cibles mentionnées comme question ouverte dans.

Plus précisément, la différence entre les attaques algébriques par canal auxiliaire et Les attaques DPA standard sont illustrées dans la figure 1. Comme déjà mentionné, les attaques DPA standard exploitent une stratégie de division et de conquête et récupèrent plusieurs morceaux d'une clé secrète indépendamment. Par exemple, dans la partie gauche de la figure, l'ensemble S1 contient typiquement les 256 candidats pour un octet clé. Afin de récupérer la bonne, l'adversaire cible une seule valeur intermédiaire (dans l'ensemble Y1). En règle générale, il peut s'agir de la sortie d'une S-box lors du premier tour de chiffrement. Chaque fuite trace li lui fournit des informations sur cette valeur intermédiaire qui est alors traduit "en informations de sous-clé. En combinant la fuite correspondant à plusieurs textes en clair (c'est-à-dire en augmentant la complexité des données q), il identifie finalement l'octet clé exactement. En revanche, un canal auxiliaire algébrique

L'attaque vise à limiter la complexité des données à q = 1 et exploite plusieurs valeurs intermédiaires (nv) dans une seule trace de fuite. Ces informations sont ensuite combinées dans une étape d'analyse cryptographique hors ligne afin de récupérer immédiatement la clé principale. Remarque que la complexité des données n'est pas toujours équivalente au nombre de mesures car une même trace de fuite peut être mesurée plusieurs (nr) fois. Mais le la complexité des données est la quantité la plus pertinente à comparer à partir d'un cryptanalytique point de vue.

Travaux connexes. Les résultats suivants peuvent être liés à trois lignes différentes De la recherche. Tout d'abord, ils visent à récupérer des informations partielles d'un appareil qui fuit de la manière la plus efficace. Ils exploitent par conséquent des techniques telles que les attaques par modèle et les modèles stochastiques. Deuxièmement, ils tirent parti de la cryptanalyse algébrique dans le cadre de la boîte noire, introduite par Courtois et Pieprzyk. En particulier, nous exploitons des solutions basées sur des solveurs SAT. Finalement, plusieurs autres articles ont suggéré de combiner attaques par canal auxiliaire avec une analyse cryptographique classique. Le problème le plus étudié est probablement celui des attaques par collision par canal auxiliaire, détaillé par ex. dans [14, 25, 26]. Techniques empruntées aux attaques carrées [6] et à la cryptanalyse différentielle [12] contre des chiffrements par blocs ont également été proposés en 2005 et 2006, respectivement. Plus récemment, des attaques par collision impossibles et multi-ensembles ont été présentées au CHES 2007 [1]. Toutes ces attaques ont des objectifs similaires aux nôtres. Ils essaient généralement de exploiter les fuites d'informations pendant plus que les premiers tours de chiffrement par blocs avec cryptanalyse avancée. Le but est de casser les implémentations pour

lesquelles ces rounds seraient protégés contre les attaques par canal auxiliaire ou pour réduire le nombre de mesures nécessaires pour effectuer une récupération de clé. Nous mentionnons enfin les attaques récentes et très efficaces basées sur les collisions de [5] qui utilisent également l'algébrique techniques et donc étroitement liées au présent article. En fait, notre proposition de cryptanalyse peut être considérée comme une généralisation d'une telle attaques. Nous visons de même à réduire la complexité des données ([5] trouvé $4 \leq q \leq 20$, nous affirmons $q = 1$). La principale différence est que nous ne sommes pas limités à un type d'information particulier (c.-à-d. Les collisions) et ne sommes pas limités à l'exploitation des premier / dernier tours d'un chiffrement par bloc. En principe, nos attaques algébriques peuvent profiter de toute fuite d'informations, de n'importe quelle partie d'un calcul cryptographique . Une conséquence est qu'ils peuvent être facilement étendus à protégés implémentations (par exemple masquées), contrairement à celles basées sur les collisions [4].

Dans cette partie, nous visons à évaluer l'impact de divers paramètres sur leur efficacité et de les appliquer à une mise en œuvre pratique de l'AES Rijndael. Par conséquent,cette section ne fournit qu'une description de haut niveau des différentes phases d'attaque.

Phase 1 hors ligne : construction du système d'équations

Le but de cette première phase est de transformer l'AES en un grand système de basse équations booléennes en degrés. Dans ce système, les bits de clé apparaissent comme des variables dans une façon dont résoudre le système équivaut à les récupérer. Dans cet article, nous exploiter les techniques présentées dans [2] pour construire notre système d'équations. dans le cas de l'AES Rijndael avec texte clair et clé de 128 bits, il en résulte un système d'environ 18 000 équations en 10 000 variables (27 000 monômes).

Phase en ligne : extraction des informations physiques

Puisque résoudre directement le système d'équations représentant l'AES Rijndael est généralement trop difficile avec les techniques actuelles, l'idée des attaques algébriques par canal auxiliaire est d'alimenter ce système avec des informations supplémentaires. Tout naturellement, physique les fuites sont de très bons candidats pour ces informations supplémentaires. Comme détaillé dans l'introduction de cet article, la question est alors de décider de ce qu'il faut extraire la mise en œuvre de la cible (une décision quelque peu arbitraire). Le plus physique informations extraites, plus la résolution est facile et plus les techniques algébriques sont intéressantes par rapport au DPA standard. Par conséquent, cela soulève deux questions :

Quelles opérations intermédiaires cibler ? Cette question dépend principalement sur la mise en œuvre cible. Par exemple, nos expériences suivantes considèrent l'AES Rijndael avec une clé principale de 128 bits dans un microcontrôleur PIC

8 bits. Dans ce contexte, SubBytes sera généralement implémenté comme une recherche de table de 256 octets et MixColumn exploitera la description de [13] dans laquelle il est implémenté avec quatre recherches de table de 256 octets et 9 opérations XOR (donnant 13 points de fuite potentiels). Il est donc naturel de profiler l'appareil cible de telle manière que les fuites correspondant à tous ces calculs intermédiaires (en plus compte tenu des ajouts clés) sont exploités par l'adversaire.

Quelles informations récupérer de chaque opération cible ? Une fois la les opérations cibles ont été décidées par l'adversaire, il reste à déterminer quoi apprendre à leur sujet. Cela dépend à nouveau de l'appareil cible (et contre-mesures qui pourraient éventuellement être incluses dans la mise en œuvre). Pour exemple, une implémentation non protégée d'AES Rijndael dans le PIC est telle que pour chacune des opérations mentionnées précédemment, les données de sortie doivent faire la navette sur un bus 8 bits. Pendant ce cycle d'horloge, les fuites seront fortement corrélé au poids de Hamming de ces données de sortie. Par conséquent, il est à nouveau naturel pour effectuer une attaque de modèle de sorte que ces poids de Hamming soient récupérés.

Surtout, les attaques algébriques par canal auxiliaire exploitent les fuites de plusieurs ciblez les opérations à la fois. Cela implique qu'il faut récupérer une information correcte sur toutes ces opérations à la fois. En effet, introduire de fausses informations dans le système d'équations conduira généralement à des incohérences et empêchera sa bonne résolution. Heureusement, cela peut être réalisé pour notre appareil cible. Comme une illustration, la partie gauche de la figure 1 illustre les traces de fuite de différentes valeurs naviguant sur le bus PIC et leur moyenne pour différents Hamming poids. Dans la partie droite de la même figure, nous avons tracé la probabilité que le Le poids de Hamming de grandes quantités d'octets cibles peut être récupéré avec confiance, exploitant éventuellement la détection d'erreur simple (ED) et l'évaluation de la probabilité (LR) (NEDLR signifie que nous n'utilisons pas ces techniques).

- La détection d'erreur consiste à rejeter les informations de canal secondaire qui à des valeurs d'entrée et de sortie incohérentes pour les S-box.

- L'évaluation de la probabilité signifie que nous n'utilisons qu'un sous-ensemble de tous les poids de Hamming valeurs extraites avec les modèles, en commençant par les plus probables.

Nous insistons sur le fait que toutes nos attaques utilisent une complexité de données de q = 1. Autrement dit, nous n'utilisons que le cryptage d'un seul texte en clair pour effectuer nos mesures. Cette complexité des données est différente du nombre de traces car on peut parfois répéter la même mesure. Dans nos expériences, nous utilisons une répétition nombre de $nr = 1$ ou 2. Ceci est très différent des attaques DPA standard qui nécessitent généralement une complexité de données beaucoup plus élevée (généralement,$10 \leq q \leq 100$).

Nous voyons qu'en utilisant ces techniques, environ 200 poids de Hamming peuvent être récupérés avec l'observation d'un seul texte en clair chiffré et cela en répétant

la mesure du même cryptage, ce nombre peut aller jusqu'à 700.

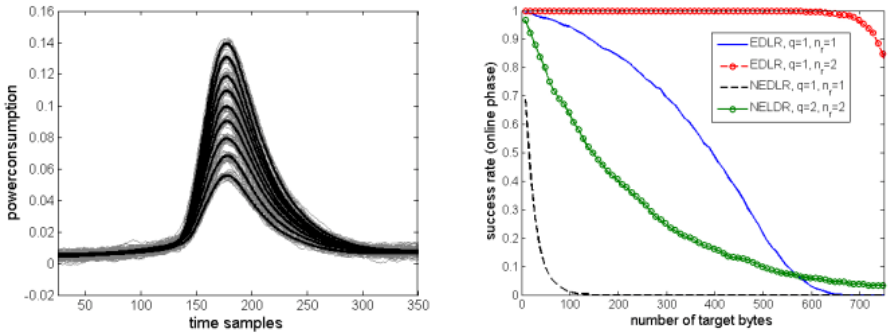


FIGURE *1.2 – taux de réussite sur plusieurs octets.*

Comme mentionné précédemment, la mise en œuvre de l'AES-128 que nous attaquons a été profilé de telle manière que nous avons récupéré les poids de Hamming correspondant à AddRoundKey (16 poids), SubBytes (16 poids) et MixColumn (4 * 13 poids dans notre mise en œuvre de l'AES). Pour les 10 tours de chiffrement, cela correspond à un maximum de 788 poids de Hamming corrects (pas toujours nécessaires).

Il est essentiel de noter que les attaques algébriques par canal auxiliaire en général pourraient fonctionner avec des choix totalement différents pour les opérations cibles et les fuites. Comme preuve de concept et en raison de leur large utilisation et de leur bonne connexion avec notre cible appareil, nous avons extrait les poids de Hamming de nos traces de consommation d'énergie. Mais on peut théoriquement exploiter tout type de fuite d'informations. Un problème ouvert particulièrement intéressant est d'appliquer ces attaques à des technologies de circuits avancées avec des comportements de fuite plus complexes. Cela pose la question de savoir comment extraire au mieux des informations partielles qui sont à la fois significatives (i.e. permet de résoudre le système) et robuste (c.-à-d. peut être facilement récupéré avec une grande confiance).

Phase 2 hors ligne : résolution du système

Une fois que le système d'équation comprenant les informations supplémentaires du canal auxiliaire est écrit, il reste à tenter de le résoudre. Différentes solutions ont été proposées dans la littérature à cet effet. L'attaque originale de Courtois et Pieprzyk proposé des techniques de linéarisation appelées XL ou XSL [10]. Basé sur la base de Groebner des techniques ont ensuite été suggérées comme alternatives possibles, par ex. dans [6, 12]. Encore une autre possibilité

est d'utiliser un solveur SAT comme dans [11]. Dans cet article, nous prenons avantage de cette dernière solution. Cela implique que le système doit être exprimé comme un problème de satisfiabilité. Le problème de satisfiabilité est la référence NP-complète problème et est largement étudié (voir [14] pour une enquête). Il consiste à déterminer si une formule booléenne (une formule combinant des variables booléennes, AND, OR et NOT gates) peut être «satisfait», c'est-à-dire s'il existe au moins une affectation des variables de telle sorte que toute la formule est vraie. La plupart des solveurs SAT nécessitent un type particulier de formule désignée comme une forme normale conjonctive (CNF). Un CNF est une conjonction (AND) de clauses, chaque clause étant une disjonction (OR) de littéraux.

En pratique, nous pouvons écrire un CNF à partir de notre système d'équations afin que le seule l'affectation valide correspond à la solution du système, comme détaillé

en 1]. Mais cette conversion en formule booléenne peut être effectuée dans un certain nombre de façons. Par exemple, les cases de substitution de l'AES peuvent être écrites sous forme d'équations booléennes non linéaires (lors de la première phase hors ligne de l'attaque) qui sont ensuite convertis en une formule booléenne, ou en un ensemble de clauses qui sont introduit directement dans le CNF mondial. Alors que la première méthode semble plus complexe et introduit (beaucoup) plus de variables intermédiaires, il a donné lieu à de meilleures aboutit à certaines de nos expériences. Nous supposons que le solveur SAT est meilleur gère certaines instances matérielles de l'attaque avec une certaine redondance et quelques variables intermédiaires supplémentaires. A titre d'illustration, la première solution donne environ 120000 clauses de 4 littéraux par boîte de substitution, contre 2048 clauses de 9 littéraux pour la deuxième solution. La taille finale du CNF dérivée du système de les équations dépendent par conséquent de la stratégie de conversion. Aux Sbox précédentes, il faut ajouter les couches linéaires (MixColumn et AddRoundKey) qui produisent environ 45 000 clauses de 4 littéraux par tour. Finalement, le les informations supplémentaires sur les canaux latéraux peuvent être définies par environ 70000 clauses de jusqu'à 8 littéraux. Cela donne une formule contenant entre un minimum de 500000 et jusqu'à plusieurs millions de clauses de 1 à 9 littéraux (le solveur SAT utilisé dans nos expériences [8] n'a pas pu en traiter plus de 5 millions).

1.5.1 Attaquer l'algorithme de planification des clés AES

Avant de commencer les investigations sur les attaques algébriques par canal auxiliaire contre l'AES, une remarque préliminaire doit être faite sur les attaques contre son algorithme de planification de clé. En raison de la structure relativement simple de l'expansion clé dans Rijndael, il est possible d'écrire un système d'équations uniquement pour cette partie du algorithme et pour le résoudre avec succès avec (même une partie) des poids de Hamming recueilli de

son exécution. De telles attaques sont alors étroitement liées au SPA proposé par Mangard dans [19]. Dans ce qui suit, nous considérons par conséquent plus cas difficile dans lequel les clés rondes sont pré-calculées dans un environnement sûr.

1.5.2 Attaquer les fonctions AES Round

Dans cette section, nous évaluons la faisabilité des attaques algébriques par canal auxiliaire contre l'AES. Pour cela, nous avons envisagé plusieurs situations. Tout d'abord, nous avons attaqué une implémentation non protégée avec une paire connue de texte brut / texte chiffré.

Ensuite, nous avons étudié l'influence des textes clairs / chiffrés inconnus. Enfin nous a examiné deux schémas de masquage différents et analysé leur résistance aux

attaques algébriques. Chaque attaque exploite l'observation d'un seul cryptage trace à partir de laquelle les informations de canal auxiliaire sont extraites. De plus, le montant des informations récupérées par l'adversaire sont utilisées comme paramètre dans nos évaluations. Il est mesuré en «nombre de tours de poids de Hamming», obtenu consécutivement (c'est-à-dire que l'adversaire récupère tous les poids de Hamming de quelques tours consécutifs - les tours choisis sont choisis en commençant au milieu du bloc chiffre) ou au hasard (c'est-à-dire que l'adversaire récupère la même quantité de Hamming poids répartis aléatoirement sur les différents calculs intermédiaires).

Nous supposons d'abord qu'aucune information de canal secondaire incorrecte n'est utilisée (qui entraînerait l'échec du solveur SAT). Par conséquent, les expériences de la section 4.1, et 4.2 se concentrer uniquement sur la deuxième partie de la phase hors ligne décrite dans la section 2.3.

Ensuite, dans la section 4.4, nous discutons du compromis entre l'applicabilité de cette phase hors ligne et l'extraction des informations en ligne décrites dans la section 2.2. Notez que pour les instances difficiles de l'attaque, le solveur SAT est parfois incapable de résoudre le système dans un délai raisonnable. Nous considérons qu'une attaque a échoué chaque fois que le solveur n'a pas trouvé de solution dans les 3600 secondes.

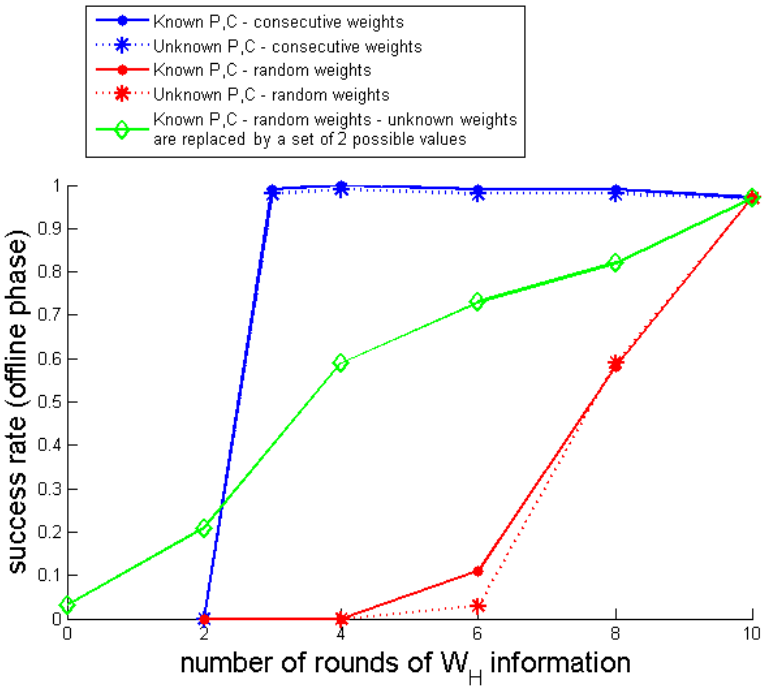
Finalement, le solveur SAT fonctionnait sur un serveur Intel avec un Xeon Processeur E5420 cadencé à 2,5 GHz exécutant un noyau Linux 32 bits 2.6.

Attaquer un appareil 8 bits avec du texte brut / chiffré connu

Les résultats de ce premier scénario sont illustrés à la figure 2 (lignes pleines). Chaque point est obtenu à partir de la moyenne sur un ensemble de 100 expériences indépendantes. Ils illustrent que le le taux de réussite des attaques

dépend fortement de la question de savoir si les fuites de canal auxiliaire correspondent à des opérations successives ou non. En effet, 3 rounds de fuites consécutives (soit 252 poids de Hamming) suffisent à résoudre plus de 95% des cas,alors que 8 séries de fuites distribuées aléatoirement ne donnent qu'un taux de réussite de 60%.

Il est intéressant de noter qu'il s'agit en fait de la fuite de l'opération MixColumn cela semble être le plus critique lors de la résolution du système. Suppression de certains les poids de Hamming obtenus à partir de cette opération ont un impact important sur l'efficacité de l'attaque. Cela peut être justifié en rappelant que MixColumn consomme une grande nombre de cycles d'horloge dans une implémentation logicielle de l'AES.



On remarque enfin que le solveur SAT peut également traiter des fuites moins précises.

Par exemple, si un poids de Hamming ne peut pas être déterminé exactement, il est possible de considérer une paire de poids de Hamming incluant le bon et d'ajouter ceci informations au système. Ce contexte est étudié dans la figure 2 (ligne encerclée).

Attaquer un périphérique 8 bits avec un texte brut / chiffré inconnu

Les lignes pointillées de la figure 2 représentent les résultats d'une attaque avec des texte clair et texte chiffré. C'est un scénario intéressant à considérer puisque classique Les attaques DPA telles que [17] échoueraient généralement à récupérer les clés dans ce contexte.

Intuitivement, c'est aussi un cas beaucoup plus difficile, car le texte en clair et le texte chiffré représentent 256 variables inconnues dans notre problème SAT. Mais à part une légère réduction du taux de réussite, nos résultats montrent que les attaques algébriques par canal auxiliaire contre l'AES Rijndael sont assez fortement immunisés contre des intrants inconnus. Ceci peut être compris en considérant que 3 tours consécutifs de Hamming le poids suffit pour récupérer le passe-partout complet.

1.6 Rappels sur les corps finis :

1.6.1 Introduction

Les bases de Gr¨obner sont un outil fondamental de l'algèbre commutative pour l'etude de systèmes polynomiaux. Généralisation de la division Euclidienne, de l'algorithme d'Euclide pour le calcul du pgcd, de l'élimination de Gauss (pour des polynômes de degré plus élevé que le degré 1), elles permettent de résoudre de nombreux problèmes concernant les systèmes polynomiaux : appartenance à un idéal, dimension et degré de l'espace des solutions, nombre de solutions dans le cas d'un nombre fini de solutions, calcul de ces solutions, etc.

Dans ce chapitre, nous présentons cet outil et ses propriétés principales, en particulier celles dont nous nous servirons dans cette thèse. Nous énonçons les théorèmes

et propriétés sans preuve.

Résolution d'un systéme d'équations

Une des principales questions concernant les systémes d'équations polynomiales est la recherche des solutions de ce système. La définition suivante précise la notion de solution d'un système :

Définition 2. *Soit $L \supset K$ une extension de corps. La variété algébrique (affine) associée à un système d'équations $\{f_1, ..., f_m\} \subset K[x_1, ..., x_n]$ sur L est l'ensemble*

des solutions (ou zéros) de $\{f_1, ..., f_m\}$ dans L, c'est-'a-dire $V_L(f_1, ..., f_m) = \{(z_1, ..., z_n) \in L : f_i(z_1, ..., z_n) = 0 \ \forall i = 1...m\}$

On appelle ensemble des solutions d'un système la variété $V_{\overline{K}}(f_1, ..., f_m)$ ou $\overline{K}$ est la clôture algébrique de K. Un système est de dimension zéro, ou zéro-dimensionnel,

si $V_{\overline{K}}(f_1, ..., f_m)$ est fini.

Pour résoudre un système, on calcule donc $V_{\overline{K}}(f_1, ..., f_m)$. Étant donné la variété V d'un système $F = \{f_1, ..., f_m\}$, les éléments de V annulent aussi tous les polynômes de

l'ensemble $\{f_1, ..., f_m\} = \{\sum f_i h_i | (h_1, ..., h_m) \in K[x_1, ..., x_n]\}$

L'ensemble $I = \langle f_1, ..., f_m \rangle$ est un idéal et on parlera d'idéal engendré par F. On le notera aussi $I = Id(F)$. Ainsi, F est un ensemble générateur de l'idéal I

Les bases de Grôbner constituent une premiére étape vers la résolution d'un systéme. Une base de Grôbner donne directement des renseignements sur les solutions du systéme dans K : dimension de la variété, nombre de solutions s'il y en a un nombre fini, degré de la variété, etc.

Trouver les solutions dans L d'un systéme de polynômes de $K[x_1, ..., x_n]$ avec $L \supset K$ une extension de corps distincte de la clôture algébrique de K constitue un probléme qui peut être difficile, comme trouver les solutions réelles d'un systéme 'a coefficients rationnels. Dans le cas particulier d'un systéme 'a coefficients dans un corps fini, il existe une méthode générale algébrique pour trouver les solutions dans F_q du systéme : en effet, les éléments du corps fini F_q sont exactement les solutions du polynôme $x_q - x$. Cette équation s'appelle équation de corps, c'est l'équation qui caractérise l'appartenance au corps F_q. Ainsi, $V_{F_q}(f_1, ..., f_m)$ est exactement l'ensemble des solutions du systéme de départ auquel on a rajouté les équations de corps. L'idéal considéré devient $I = \langle f_1, ..., f_m, x_1^q - x_1, ..., x_n^q - x_i^n \rangle$, et on a donc la relation :

$V_{F_q}(f_1, ..., f_m) = V_{F_q}(f_1, ..., f_m, x_1^q - x_1, ..., x_n^q - x_i^n)$

De plus, l'idéal I a de bonnes propriétés : grâce aux équations de corps, l'idéal est radical1 et posséde un nombre fini de solutions .

1.6.2 Les Corps finis ou corps de Galois $GF(2^8)$

L'ensemble $Z_p = \{0, \cdots, p-1\}$, noté Z/p, muni des opérations d'addition et de multiplication modulo p forme un corps fini si et seulement si p est premier.

En l'honneur du mathématicien français Évariste Galois, fondateur de la théorie des corps finis, ce corps est nommé corps de Galois. Il est d'ordre p et est noté $GF(p)$.

Comme un corps de Galois est en fait l'anneau des entiers modulo p, les opérations d'addition, de soustraction et de multiplication sont réalisées comme sur un entier standard suivi de la réduction modulo p. Ainsi, dans $GF(5), 4+3=7$ est réduit à $2 modulo 5$. Enfin, la division dans un corps de Galois est la multiplication par l'inverse modulo p.

Le plus petit corps de Galois : $GF(2)$, contient les éléments 0 et 1 et nous avons $GF(2) = \{0, 1\}$. Dans $GF(2)$ l'addition est réalisée par un ou exclusif XOR et la multiplication par un AND. Enfin, comme le seul élément inversible est 1, la division est la fonction identité.

1.6.3 Construction des corps de Galois [6]

Soit $K[x]$ un anneau polynomial formé de l'ensemble de tous les polynômes à une variable x sur le corps K et muni des opérations d'addition et de multiplication polynomiales. L'anneau quotient $\frac{K[x]}{(f(x))}$ est un corps si et seulement si $f(x)$ est irréductible dans $K[x]$.

Soient K un corps fini d'ordre $q = p^n$ et $f(x) \in K[x]$ un polynôme irréductible de degré d. L'anneau quotient $A = \frac{K[x]}{(f(x))}$est un corps d'ordre $q^d = p^{nd}$ qui est une extension de degré d de K. Ces éléments peuvent être présentés sous la forme :

$c_{d-1}x^{d-1} + ... + c_2x^2 + c_1x + c_0$

avec $ci \in K$. Tout corps fini d'ordre p^{nd} est isomorphe à A.

Il existe d'autres façons de construire et de représenter les éléments d'un corps fini. Nous retiendrons que, à toute puissance première correspond un corps fini et que, par conséquent, toutes les représentations de $GF(2^8)$ sont isomorphes. Cependant, malgré cette équivalence, la représentation choisie a un impact sur la complexité de l'implémentation. Le standard de l'AES a choisi d'utiliser la représentation polynomiale.

Ainsi, un octet contenant les bits b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0 est considéré comme un polynôme avec des coefficient dans $\{0, 1\}$:

$b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x^1 + b_0$

Par exemple, le caractère alphanumérique "a" a la valeur hexadécimale 61, soit 01100001 en binaire, dans la table de correspondance ASCII. Sa traduction

polynomiale est donc la suivante :

$$x^6 + x^5 + 1$$

Opérations dans $GF(2^8)$ C'est dans le corps fini $GF(2^8)$ que l'AES effectue ses opérations. Concrètement, cela signifie que les calculs sont effectués sur des polynômes de degré 7 à coefficients dans $\{0, 1\}$, modulo un polynôme irréductible $m(x)$ de degré8 valant $x^8 + x^4 + x^3 + x + 1$ pour l'AES.

L'addition L'addition de deux éléments dans $GF(2^8)$ est réalisée en ajoutant les coefficients des puissances correspondantes des polynômes représentant ces éléments. Cet ajout s'effectuant $modulo2$, cela revient à appliquer un XOR sur les coefficients.

La multiplication La multiplication dans $GF(2^8)$ s'effectue modulo un polynôme irréductible.Nous avons vu que pour l'AES, ce polynôme est $m(x) = x^8 + x^4 + x^3 + x + 1$

La réduction modulaire permet de garantir que le résultat de la multiplication sera un polynôme de degré inférieur à 8 et donc, que ce dernier correspondra bien à la représentation d'un octet.

La multiplication de deux polynômes s'effectue en deux étapes. Dans un premier temps, la multiplication est effectuée classiquement et dans un deuxième temps, la réduction polynomiale est appliquée au résultat.

Chapitre 2

Bases de Gröbner.

Dans ce chapitre, nous allons discuter les bases de Gröbner. L'objectif principal de ce chapitre est de motiver et de présenter l'algorithme original de Buchberger pour le calcul des bases de Gröbner.

La fin de ce chapitre est une brève discussion sur la façon dont les bases de Gröbner sont utiles pour résoudre des systèmes d'équations polynomiales car c'est l'application principale de ce travail.

— "Ideals, Varieties, and Algorithms" par Cox, Little, et O'Shea[5],
— "Gröbner Bases – A Computational Approach to Commutative Algebra" par Becker et Weispfenning [2] et
— "Computational Commutative Algebra" par Kreuzer et Robbiano[12].

2.1 Notation

La notation et les conventions suivantes sont utilisées dans ce texte :

— Nous commençons à compter à zéro par défaut.
— $\mathbb{F}$ est un corps, pas nécessairement clos algébriquement. $\overline{\mathbb{F}}$ représentant la clôture algébrique de $\mathbb{F}$. Dans les listes de code source, nous utilisons généralement `K` pour dé signer le corps afin d'éviter toute confusion avec F défini ci-dessous.$\mathbb{F}_p$ est le corps fini d'ordre p avec p premier ;$\mathbb{F}_{p^n}$ le corps d'extension fini de degré n sur $\mathbb{F}_p$.
— $\mathbb{Z}$ est l'anneau d'entiers ; $\mathbb{Z}_{\geq 0}$ sont les entiers ≥ 0.
— P est un anneau polynomial $\mathbb{F}[x_0, \ldots, x_{n-1}]$ dans les variables $x_0, \ldots, x_{n-1}$.
— $F = (f_0, \ldots, f_{m-1})$ est une liste ordonnée de polynômes dans P ; on note $\{f_0, \ldots, f_{m-1}\}$ un ensemble de polynômes non ordonnés $f_0, \ldots, f_{m-1}$.

— On appelle $m = x_0^{\alpha_0} x_1^{\alpha_1} \dots x_{n-1}^{\alpha_{n-1}}$ avec $\alpha_i \in \mathbb{Z}_{\geq 0}$ un monôme et $t = c \cdot m$ avec $c \in \mathbb{F}$ un terme. Notez que certains auteurs comme [2] changent la définition des termes et monômes utilisés ici.
— Si $m = x_0^{\alpha_0} x_1^{\alpha_1} \dots x_{n-1}^{\alpha_{n-1}}$ est un monôme, on appelle $\alpha_0, \alpha_1, \dots \alpha_{n-1}$ son vecteur exposant :

$$expvec(m) = \alpha_0, \alpha_1, \dots \alpha_{n-1}.$$

— $M(f)$ est l'ensemble des monômes qui apparaissent dans le polyn ôme f et $T(f)$ l'ensemble des termes qui apparaissent dans le mê me polynôme f. Nous étendons cette définition aux ensembles de polynômes $F = f_0, \dots, f_{m-1} : M(F) = \bigcup_{i=0}^{m-1} M(f_i)$ et $T(F) = \bigcup_{i=0}^{m-1} T(f_i)$
— $\deg(m)$ est le degré du monôme m$m = x_0^{\alpha_0} x_1^{\alpha_1} \dots x_{n-1}^{\alpha_{n-1}}$ défini comme $\sum_{i=0}^{n-1} \alpha_i$. Nous étendons cette définition aux polynômes tels que $\deg(f)$ pour $f = \sum c_i m_i$ est défini comme $\max\{\deg(m_i)\}$. Nous définissons $\deg(\alpha)$ comme $\deg(m)$ pour $\alpha = expvec(m)$.
— $A[i, j]$ représente l'élément de la ligne i et de la colonne j de la matrice A.
— $f \ \% \ g$ désigne le résultat de l'opération modulo f modulo g.

Chaque fois que cela est approprié, des exemples sont fournis pour illustrer des théorèmes, des algorithmes et des propositions. De plus, si possible, des extraits de code source sont fournis pour reproduire des exemples dans le logiciel mathématique Sage.

Sage est un logiciel de mathématiques open-source qui vise à fournir une «alternative viable à Magma, Maple, Mathematica et Matlab».

Par exemple, consider the following set of polynomials in $\mathbb{F}_{127}[x, y, z]$.

$$f_0 = 81z^2 + 51x + 125z + 38$$
$$f_1 = 76xy + 80y^2 + 49xz + 62yz + 45z^2$$
$$f_2 = 122x^2 + 106yz + 78z^2 + 48x + 112y$$

Cet exemple peut être construit dans Sage comme suit :

```
sage: K = GF(127)
sage: P.<x,y,z> = PolynomialRing(K)
sage: f0 = -46*z^2 + 51*x - 2*z + 38
sage: f1 = -51*x*y - 47*y^2 + 49*x*z + 62*y*z + 45*z^2
sage: f2 = -5*x^2 - 21*y*z - 49*z^2 + 48*x - 15*y
```

2.2 ordre monomial

Lorsque nous considérons des polynômes univariés, il est simple de déterminer quel monôme est le plus grand et lequel est le plus petit. Une fois que nous

considérons les polynômes multivariés, les choses ne sont plus aussi simples. Ainsi, nous attachons un ordre de monôme ou un ordre de terme à un anneau qui code la façon dont nous comparons les monômes.

Définition 3 (ordre monomial). *Un ordre ùonomial sur* $\mathbb{F}$ *est une relation* $>$ *on* $\mathbb{Z}^n_{\geq 0}$*, ou de manière équivalente, toute relation sur l'ensemble des monômes* x^α,$\alpha \in \mathbb{Z}^n_{\geq 0}$ *, satisfaisant :*

1. $>$ *est un ordre total (ou linéaire) sur* $\mathbb{Z}^n_{\geq 0}$.
2. *si* $\alpha > \beta$ *et* $\gamma \in \mathbb{Z}^n_{\geq 0}$ *alors* $\alpha + \gamma > \beta + \gamma$.
3. $>$ *est un bon ordre sur* $\mathbb{Z}^n_{\geq 0}$. *a un plus petit élément sous* $>$. *Deux des ordres monômes les plus utilisés sont l'ordre "lexicographique" et le "degré lexicographique inversé".*

Définition 4 (Ordre lexicographique *lex*). *soit le vecteur exposant* $\alpha = (\alpha_0, \ldots, \alpha_{n-1})$ *et* $\beta = (\beta_0, \ldots, \beta_{n-1}) \in \mathbb{Z}^n_{\geq 0}$. *Nous disons que* $\alpha \underset{lex}{>} \beta$ *si, dans la diff érence de vecteur* $\alpha - \beta \in \mathbb{Z}^n$, *l'entrée non nulle la plus à gauche est positive. Nous écrirons* $x^\alpha \underset{lex}{>} x^\beta$ *si* $\alpha \underset{lex}{>} \beta$.

Nous montrerons plus loin dans ce chapitre que lex est un ordre qui permet de "lire" la solution d'un système d'équations polynomiales multivariées à partir de la base de Gröbner. C'est parce que lex est un ordre d'élimination. Mais en pratique, calculer une base lexicographique de Gröbner est généralement moins efficace que calculer une base lexicographique inversée

Définition 5 (Ordre lexicographique inversé en degrés *degrevlex*). *Soit le vecteur exposant* $\alpha = (\alpha_0, \ldots, \alpha_{n-1})$ *et* $\beta = (\beta_0, \ldots, \beta_{n-1}) \in \mathbb{Z}^n_{\geq 0}$. *on dit* $\alpha \underset{degrevlex}{>} \beta$ *si soit*

- $\deg(\alpha) > \deg(\beta)$ *or*
- $\deg(\alpha) = \deg(\beta)$ *et l'entrée non nulle la plus à droite dans la différence vectorielle* $\alpha - \beta \in \mathbb{Z}^n$ *est négatif.*

Nous écrirons $x^\alpha \underset{degrevlex}{>} x^\beta$ *si* $\alpha \underset{degrevlex}{>} \beta$.

Nous aurons également besoin d'ordres de blocs plus tard dans ce travail, qui sont des ordres d'élimination potentiellement "mélangés" avec d'autres ordres de monômes.

Définition 6 (Ordre de blocs ou de produits). *Soit* $x = (x_0, \ldots, x_{n-1})$ *et* $y = (y_0, \ldots, y_{m-1})$ *être deux ensembles ordonnés de variables,* $<_1$*un ordre monôme sur* $\mathbb{F}$ *et* $<_2$ *un ordre de monôme sur* $\mathbb{F}[y_0, \ldots, y_{m-1}]$. *On dit que* $x^a y^b < x^A y^B$ *par rapport à l'ordre de bloc (ou l'ordre de produit)* $(<_1, <_2)$ *on* $\mathbb{F}[x_0, \ldots, x_{n-1}, y_0, \ldots, y_{m-1}]$ *si soit*

- $x^a <_1 x^A$ *ou*

— $x^a = x^A$ *et* $y^b <_2 y^B$.

si Inductivement on définit l'ordre de produit de plus de deux ordres mon ômes.

Nous écrirons simplement $x > y$ s'il ressort clairement du contexte à quel ordre nous nous référons. Nous pouvons étendre l'ordre des monômes aux polynômes en comparant d'abord les plus grands monômes et en comparant les plus petits monômes uniquement s'ils sont égaux.

Par exemple, considérons le polynôme

$$f = 1 + y_0 + x_2 + x_1 + x_0 + x_0x_1 \in \mathbb{F}[y_0, x_0, x_1, x_2].$$

En ce qui concerne l'ordre lexicographique des monômes et $y_i > x_i$ nous avons que le monôme principal de f est y_0 mais en ce qui concerne le degré d'ordre lexicographique inverse, le monôme principal est x_0x_1 y_0 et x_0, x_1, x_2 et choisissons *degrevlex* dans les deux blocs, nous avons que y_0 est le monôme principal.

les ordres sont attribués aux anneaux polynomiaux multivariés dans Sage en utilisant le mot-clé `order` :

```
sage: P.<y0,x0,x1,x2> = PolynomialRing(QQ, order='lex')
sage: f = 1 + y0 + x2 + x1 + x0 + x0*x1
sage: f.lm()
y0

sage: P.<y0,x0,x1,x2> = PolynomialRing(QQ, order='degrevlex')
sage: f = 1 + y0 + x2 + x1 + x0 + x0*x1
sage: f.lm()
x0*x1

sage: T = TermOrder('degrevlex',1) + TermOrder('degrevlex',3)
sage: P.<y0,x0,x1,x2> = PolynomialRing(QQ, order=T)
sage: f = 1 + y0 + x2 + x1 + x0 + x0*x1
sage: f.lm()
y0
```

Nous désignons le plus grand monôme d'un polynôme f comme le monôme principal $LM(f)$, son coefficient comme le coefficient principal $LC(f)$ et leur produit comme terme principal $LT(f) = LC(f) \cdot LM(f)$.

2.3 Base de Gröbner

Nous nous intéressons aux idéaux des anneaux polynomiaux multivariés et à leurs bases.

Définition 7 (Idéale). *Un sous-ensemble $I \subset P$ est un idéal s'il satisfait :*

1. $0 \in I$;
2. *si* $f, g \in I$, *alors* $f + g \in I$;
3. *si* $f \in I$ *et* $h \in P$, *alors* $h \cdot f \in I$.

Définition 8. *Soit $f_0, \ldots, f_{m-1}$ être des polynômes dans P . Définir l'ensemble*

$$\langle f_0, \ldots, f_{m-1} \rangle = \left\{ \sum_{i=0}^{m-1} h_i f_i : h_0, \ldots, h_{m-1} \in P \right\}.$$

Cet ensemble I est un idéal appelé l'idéal généré par $f_0, \ldots, f_{m-1}$.

Définition 9 (Leading Monomial Ideal). *Soit I être un idéal $\subset P$ et définir l'ensemble*

$$\{LM(f_i) \mid f_i \in I\}.$$

Nous appelons l'idéal englobé par cet ensemble l'idéal monô me principal de I et le désignons par $\langle LM(I) \rangle \subset P$.

S'il existe un ensemble fini de polynômes dans P qui génère un idéal donné, cet ensemble est appelé une base. Le théorème de base de Hilbert stipule que tout idéal de P est généré de manière finie :

Théorème 10 (Théorème de base de Hilbert). *Chaque idéal $I \subset P$ a un groupe gén érateur fini. Autrement dit, $I = \langle f_0, \ldots, f_{m-1} \rangle$ pour certain $f_0, \ldots, f_{m-1} \in I$.*

Proof. Voir [5, p. 74].

Notez que la plupart des idéaux ont de nombreuses bases différentes.

Le théorème de base de Hilbert a des conséquences importantes pour les calculs de base de Gröbner. . La première est qu'une séquence croissante imbriquée d'idéaux $I_0 \subset I_1 \subset \ldots$ on P se stabilise à un certain moment. Explicitement :

Théorème 11 (Condition de la chaîne ascendante). *Soit*

$$I_0 \subset I_1 \subset I_2 \subset \ldots$$

une chaîne ascendante d'idéaux dans P. Alors il existe un $N \geq 1$ tel que

$$I_N = I_{N+1} = I_{N+2} = \ldots \ .$$

Proof. Voir [5, p.76].

Définition 12 (Anneau noéthérien)**.** *Un anneau pour lequel la condition de chaîne ascendante pour les idé aux tient est appelé un anneau noetherian.*

Les bases de Gröbner sont définies comme suit :

Définition 13 (Bases de Gröbner)**.** *Soit I un idéal de $\mathbb{F}$ et fixe un ordre monôme. Un sous-ensemble fini*

$$G = \{g_0, \ldots, g_{m-1}\} \subset I$$

est dit être une base de Gröbner ou une base standard de I si

$$\langle LM(g_0), \ldots, LM(g_{m-1})\rangle = \langle LM(I)\rangle.$$

Ainsi pour tout $f_i \in I$ nous avons que $LM(f_i))$ est divisible par un certain $LM(g_i) \in G$.

Exemple 14. *Par exemple, une base de Gröbner par rapport à l'ordre des monô mes* degrevlex *et $x > y > z$ pour l'exemple présenté précé demment*

$$\begin{aligned} f_0 &= 81z^2 + 51x + 125z + 38 \\ f_1 &= 76xy + 80y^2 + 49xz + 62yz + 45z^2 \\ f_2 &= 122x^2 + 106yz + 78z^2 + 48x + 112y \end{aligned}$$

est :

$$\begin{aligned} g_0 &= y^3 + 66y^2z + 72y^2 + 98xz + 64yz + 56x + 16y + 38z + 53, \\ g_1 &= x^2 + 55yz + 99x + 3y + 57z + 60, \\ g_2 &= xy + 108y^2 + 9xz + 71yz + 57x + 65z + 35, \\ g_3 &= z^2 + 90x + 116z + 82. \end{aligned}$$

Cette base de Gröbner a été obtenue par la séquence de commandes suivante dansSage :

```
sage: P.<x,y,z> = PolynomialRing(GF(127),order='degrevlex')
sage: f = -46*z^2 + 51*x - 2*z + 38
sage: g = -51*x*y - 47*y^2 + 49*x*z + 62*y*z + 45*z^2
sage: h= -5*x^2 - 21*y*z - 49*z^2 + 48*x - 15*y
sage: I = Ideal(f,g,h)
sage: I.groebner_basis()
[y^3 - 61*y^2*z - 55*y^2 - 29*x*z - 63*y*z + 56*x + 16*y + ...,
x^2 + 55*y*z - 28*x + 3*y + 57*z + 60,
x*y - 19*y^2 + 9*x*z - 56*y*z + 57*x - 62*z + 35,
z^2 - 37*x - 11*z - 45]
```

Définition 15 ([5]). *Fixons un ordre de monôme $>$ sur $\mathbb{Z}^n_{\geq 0\_}$ et soit $F = (f_0, \ldots, f_{s-1})$ un s-tuple ordonné de polynômes dans $\mathbb{F}$. Alors chaque $f \in \mathbb{F}$ peut être écrit comme*

$$f = a_0 f_0 + \cdots + a_{s-1} f_{s-1} + r,$$

où $a_i, r \in \mathbb{F}$ et $r = 0$ ou r est une combinaison linéaire, avec des coefficients dans $\mathbb{F}$, de monômes, dont aucun n'est divisible par l'un des $LM(f_0), \ldots, LM(f_{s-1})$. On appelle r reste *de f sur la division par $\mathbb{F}$. De plus, si $a_i f_i \neq 0$, alors on a*

$$expvec(LM(f)) \geq expvec(LM(a_i f_i)).$$

Nous écrivons

$$\overline{f}^F = r.$$

Proof. Voir [5, p.62ff].

Un algorithme pour calculer $a_0, \ldots, a_{s-1}$ est donné dans l'algorithme 1.

Input : $(f_0, \ldots, f_{s-1}, f)$ – a $s+1$-tuple of polynomials $\in P$.
Result : $a_0, \ldots, a_{s-1}, r$ – a $s+1$-tuple of polynomial $\in P$.
begin
  $a_i \longleftarrow 0$; $r \longleftarrow 0$; $p \longleftarrow f$;
  while $p = 0$ **do**
    $i \longleftarrow 0$;
    $divisionoccured \longleftarrow False$;
    while $i < s$ **and** $divisionoccured = False$ **do**
      if $\mathrm{LT}(f_i) \mid \mathrm{LT}(p)$ **then**
        $a_i \longleftarrow a_i + \mathrm{LT}(p)/\mathrm{LT}(f_i)$;
        $p \longleftarrow p - \mathrm{LT}(p)/\mathrm{LT}(f_i) f_i$;
      end
      else
        $i \longleftarrow i + 1$;
      end
    end
    if $divisionoccured = False$ **then**
      $r \longleftarrow r + \mathrm{LT}(p)$;
      $p \longleftarrow p - \mathrm{LT}(p)$;
    end
  end
  return $a_0, \ldots, a_{s-1}, r$;
end

Algorithmus 1 : Long Division

Les bases de Gröbner ont plusieurs propriétés intéressantes : le reste r de la division de tout

$f \in P$ par G est unique et les bases de Gröbner réduites sont une représentation unique d'un idéal par rapport à un ordre monôme.

Définition 16 (Base de Gröbner réduite)**.** *Une base de Gröbner réduite pour un idéal polynomial I est une base de Gröbner G telle que :*

1. $LC(f) = 1$ *pour tous* $f \in G$ *;*
2. $\forall f \in G, \nexists m \in M(f)$ *tele que* $m \in \langle LM(G \setminus \{f\})\rangle$.

Par défaut, Sage calcule toujours la base de Gröbner réduite lors du calcul d'une base de Gröbner. Si une base de Gröbner a été obtenue par d'autres moyens, la fonction

```
MPolynomialIdeal.interreduced_basis()
```

peut être

Notez que Sage - contrairement à d'autres systèmes comme SINGULAR fait la différence entre les tuples de polynômes et les idéaux. Les idéaux sont des objets de premier ordre dans Sage.

2.4 Algorithme de Buchberger

En 1965, Bruno Buchberger a introduit la notion de base de Gröbner et aussi un critère pour tester si un ensemble de polynômes est une base de Gröbner. Ce critère conduit naturellement à l'algorithme de Buchberger pour calculer une base de Gröbner à partir d'une base idéale donnée. Les principaux concepts de son critère sont présentés ci-dessous.

Considérons un ensemble de polynômes $G = \{f_0, \ldots, f_{m-1}\} \subset \mathbb{F}$. S'il existe un $m \in \langle LM(I)\rangle$ avec

$$m \notin \langle LM(f_0),\ \ldots\ , LM(f_{m-1})\rangle,$$

alors G n'est pas une base de Gröbner pour $\langle G\rangle$; cela découle de la définition des bases de Gröbner.

Afin d'obtenir un candidat pour un tel m, on peut choisir deux éléments f_i et f_j de G et calculer

$$s = ax^{\alpha} f_i - bx^{\beta} f_j.$$

On sait que $LM(ax^{\alpha} f_i - bx^{\beta} f_j) \in \langle LM(I) \rangle$ car $ax^{\alpha} f_i - bx^{\beta} f_j \in I$. Supposons maintenant que dans les termes $ax^{\alpha} LT(f_i)$ et $bx^{\beta} LT(f_j)$ $(a, b \in \mathbb{F})$ s'annulent. Si en conséquence $LM(ax^{\alpha} f_i - bx^{\beta} f_j)$ n'est pas dans l'idéal $\langle LM(f_0), \ldots, LM(f_{t-1}) \rangle$ nous savons que G ne peut pas être une base de Gröbner.

Les S-polynômes sont une (en fait : la) façon de construire les annulations requises des termes principaux :

Définition 17 (S-polynômes)**.**

Soit $f, g \in \mathbb{F}$ être des polynômes non nuls.

1. *si $\alpha = expvec(LM(f))$ et $\beta = expvec(LM(g))$ alors soit $\gamma = (\gamma_0, \ldots, \gamma_{n-1})$ où $\gamma_i = \max(\alpha_i, \beta_i)$ $foreveryi < n$. On a alors que x^{γ} est le plus petit commun multiple de $LM(f)$ et $LM(g)$, écrit comme*

$$x^{\gamma} = LCM(LM(f), LM(g)).$$

2. *Le S-polynôme de f et g est défini comme*

$$S(f, g) = \frac{x^{\gamma}}{LT(f)} \cdot f - \frac{x^{\gamma}}{LT(g)} \cdot g.$$

On appelle f et g les générateurs de $S(f, g)$ et $\frac{x^{\gamma}}{LT(f)} \cdot f$ et $\frac{x^{\gamma}}{LT(g)} \cdot g$ les composantes de $\frac{x^{\gamma}}{LT(g)} \cdot g$. Parfois, il est avantageux de considérer les produits des composants comme non évalués, à savoir comme les tuples $(\frac{x^{\gamma}}{LT(f)}, f)$ et $(\frac{x^{\gamma}}{LT(g)}, g)$.Nous appelons le tuple (f, g) une paire critique

L'exemple suivant montre que $S(f, g)$ est construit de manière à permettre l'annulation des termes principaux.

Exemple 18. *Soit $f_0 = x^3 - 2xy$ et $f_1 = x^2y - 2y^2 + x$. Les monômes principaux par rapport au* degrevlex *et $x > y$ sont $LM(f_0) = x^3$ et $LM(f_1) = x^2y$ Et ainsi $x^{\gamma} = x^3y$. Le S-polynôme est :*

$$\begin{aligned}
S(f_0, f_1) &= \frac{x^{\gamma}}{LT(f_1)} \cdot f_1 - \frac{x^{\gamma}}{LT(f_2)} \cdot f_2 \\
S(f_0, f_1) &= \frac{x^3y}{x^3} \cdot (x^3 - 2xy) - \frac{x^3y}{x^2y} \cdot (x^2y - 2y^2 + x) \\
S(f_0, f_1) &= y \cdot (x^3 - 2xy) - x \cdot (x^2y - 2y^2 + x) \\
S(f_0, f_1) &= x^3y - 2xy^2 - x^3y + 2xy^2 - x^2 \\
S(f_0, f_1) &= -x^2
\end{aligned}$$

Le même exemple dans Sage :

```
sage: P.<x,y> = PolynomialRing(QQ, order='degrevlex')
sage: f0 = x^3 - 2*x*y
sage: f1 = x^2*y -2*y^2 + x
sage: (x^3*y)//x^3 * f0 - (x^3*y)//(x^2*y) * f1
-x^2
```

L'éducation `sage.rings.polynomial.toy_buchberger` Le module propose également une fonction `spol` :

```
sage: from sage.rings.polynomial.toy_buchberger import spol
sage: spol(f0,f1)
-x^2
```

Le lemme suivant indique que chaque fois que des combinaisons de termes s'annulent dans un polynôme, cette annulation peut être expliquée en S-polynômes.

Lemme 19. *Soit le terme principal de chaque sommation de*

$$s = \sum_{i=0}^{t-1} c_i x^{\alpha_i} g_i \in \mathbb{F}$$

avec $c_i \in \mathbb{F}$*, avoir le vecteur exposant*

$$\delta = \alpha_i \ + \ expvec(LM(g_i)) \in \mathbb{Z}^{\ltimes}_{\geq \nvdash} \ if \sqsupset \ \neq \ \nvdash.$$

si $expvec(LM(s))$ *est plus petite que* δ*, alors* s *est une combinaison linéaire des S-polynômes* $S(f_j, f_k)$ *pour* $0 \leq j, k < t$ *avec coefficients* c_i *dans* $\mathbb{F}$*. En outre, chaque monôme principal de* $S(f_j, f_k)$ *a un vecteur exposant* $< \delta$*.*

Proof. Voir [5, p.81ff].

L'idée clé du critère constructif de Buchberger pour les bases de Gröbner est d'utiliser ces S-polynômes pour construire de nouveaux éléments $S(f, g)$ dans l'idéal avec un terme principal plus petit que ceux des composantes de $S(f, g)$. Si de tels éléments peuvent être trouvés dont les termes principaux ne sont pas des multiples des termes principaux d'autres éléments déjà dans la base, alors la base n'est pas une base de Gröbner.

Théorème 20 (Critère de Buchberger)**.** *Soit* I *être un idéal.* $G = \{g_0, \ \ldots \ , g_{s-1}\}$ *est une base de Gröbner pour* I*, si et seulement si pour toutes les paires* $i \neq j$*, le reste* r *de la division de* $S(g_i, g_j)$ *par* G *(listée dans un certain ordre) est z éro, c'est-à-dire que nous avons* $\overline{f}^G = 0$*.*

Proof. Voir [5, p.82ff]

Exemple 21. *Soit $f_0 = x^3 - 2xy$ et $f_1 = x^2y - 2y^2 + x$.Le S-polynôme est $-x^2$ qui n'est réductible ni par $LM(f_0) = x^3$ ni par $LM(f_1) = x^2y$. Ainsi, (f_0, f_1) n'est pas une base de Gr öbner.*

Il existe un autre critère - lié - qui peut être vérifié pour vérifier si un ensemble donné de polynômes forme une base de Gröbner ou non. Pour ce critère, l'expression f se réduit à zéro modulo G est nécessaire.

Définition 22. *[5, p.100] Fixez un ordre monôme et soit $G = \{g_0, \ldots, g_{s-1}\} \subset P$ un ensemble non ordonné de polynômes. Étant donné un polynôme $f \in P$, on dit que f se réduit à zéro modulo G, écrit*

$f \underset{G}{\longrightarrow} 0$,*si f peut être écrit sous la forme*

$$f = a_0 g_0 + \cdots + a_{s-1} g_{s-1},$$

avec $a_i \in P$ tel que chaque fois que $a_i g_i \neq 0$, on a

$$LM(f) \geq LM(a_i g_i).$$

Alternativement, nous pouvons exprimer ce concept en utilisant la notion de t-représentations :

Définition 23 (t-Représentation)**.** *Fixez un ordre monôme et soit $G = \{g_0, \ldots, g_{s-1}\} \subset P$ un ensemble non ordonné de polynômes et soit t un monôme. Étant donné un polynôme $f \in P$, on dit que f a une t-repré sentation si f peut s'écrire sous la forme*

$$f = a_0 g_0 + \cdots + a_{s-1} g_{s-1},$$

tel que chaque fois que $a_i g_i \neq 0$, nous avons $a_i g_i \leq t$. De plus, nous avons que $f \underset{G}{\longrightarrow} 0$

si et seulement si f a une $LM(f)$-représentation par rapport à G.

Veuillez noter que $\overline{f}^G = 0$ implique $f \underset{G}{\longrightarrow} 0$ mais l'inverse ne tient pas en général [5, 100ff].

Exemple 24. *Considérons un certain $\mathbb{F}[x, y]$ avec l'ordre monôme lexicographique inverse de degré et $f = xy^2 - x$ et $G = (xy + 1, y^2 - 1)$.. L'algorithme de division (cf. Algorithme 1) donne $f = xy^2 - x = y \cdot (xy+1) + 0 \cdot (y^2-1) + (-x-y)$ ce qui implique $\overline{f}^G \neq 0$.Par contre on peut écrire $f = xy^2 - x = 0 \cdot (xy+1) + x \cdot (y^2-1)$ ce qui implique $f \underset{G}{\longrightarrow} 0$.*

En utilisant cette définition, le critère de Buchberger peut être reformulé comme suit :

Théorème 25. *Une base $G = \{g_0, \ldots, g_{s-1}\}$ pour un idéal I est une base de Gröbner si et seulement si*

$$S(g_i, g_j) \underset{G}{\longrightarrow} 0$$

pour tous $i \neq j$.

La preuve de ce théorème découle directement de la preuve du critère de Buchberger dans [5].

Le critère de Buchberger (théorème 20) et la condition de chaîne ascendante (théorème 11) conduisent à l'algorithme suivant pour calculer la base de Gröbner :

```
Input : F – a finite subset of P
Result : a Gröbner basis for the ideal spanned by F
begin
    G ⟵ F;
    G_2 ⟵ ∅;
    while G_2 ≠ G do
        G_2 ⟵ G;
        for f, g ∈ G_2 × G_2 do
            if LM(f) < LM(g) then
                s̃ ⟵ S(f, g)^G;
                if s̃ ≠ 0 then add s̃ to G;
                ;
            end
        end
    end
    return G;
end
```

Algorithmus 2 : Buchberger's Algorithm

L'exactitude et la terminaison de cet algorithme peuvent être dérivées des trois observations suivantes :

1. A chaque étape de l'algorithme, $G \subset I$ et $\langle G \rangle = I$.je tiens.
2. Si $G_2 = G$ alors $S(f, g) \underset{G}{\longrightarrow} 0$ pour tout $f, g \in G$ et, d'après le critère de Buchberger, G est une base de Gröbner.
3. L'égalité $G_2 = G$ se produit par étapes finies puisque les idéaux $\langle LM(G) \rangle$, à partir d'it érations successives de la boucle, forment une chaîne ascendante. En raison de la condition de chaîne ascendante (théorème11) cette chaîne d'idéaux se stabilise après un nombre fini d'itérations et à ce moment $\langle LM(G) \rangle = \langle LM(G_2) \rangle$ tient, ce qui implique $G_2 = G$.
   Une implémentation simple de l'algorithme de Buchberger dans Sage est donnée ci-dessous :

```
spol = lambda f,g: LCM(f.lm(),g.lm())//f.lt()*f - \
LCM(f.lm(),g.lm())//g.lt()*g
def buchberger(F):
G = set(F)
G2 = set()
while G2!=G:
G2 = copy(G)
for f,g in cartesian_product_iterator([G2,G2]):
if f<g:
s = spol(f,g).reduce(G2)
if s != 0:
G.add(s)
return G
```

Il est implémenté en tant que `buchberger` dans `sage.rings.polynomial.toy_buchberger` par l'auteur :

```
sage: P.<x,y> = PolynomialRing(QQ, order='degrevlex')
sage: f0 = x^3 - 2*x*y
sage: f1 = x^2*y -2*y^2 + x
sage: buchberger(Ideal(f0,f1))
[x^2*y - 2*y^2 + x, -2*x*y, -2*y^2 + x, x^2, x^3 - 2*x*y]
```

Même si cet algorithme se termine finalement, il est bien connu [2, p.511ff] que son exécution n'est pas polynomiale en nombre de variables, car les bases intermédiaires G_2 croissent de façon exponentielle au cours des calculs. En particulier, nous avons le théorème suivant :

Théorème 26 ([8]). *Soit I un idéal dans $\mathbb{F}_q[x_0, \ldots, x_{n-1}]$ généré par des polynômes $f_0, \ldots, f_{n-1}$ de degré s $d_0, \ldots, d_{n-1}$ respectivement. Supposons que l'idéal I soit de dimension zéro (défini dans la section 2.5).*

- *Un calcul basé sur Gröbner pour un ordre monôme* lexicographical *atteint au plus le degré $D \leq \prod_{i=0}^{n-1} d_i$.*
- *Un calcul basé sur Gröbner pour un ordre monôme lexicographique inversé de degré atteint au plus le degré $D \leq 1 - n + \sum_{i=0}^{n-1} d_i$.*
  L'algorithme de Buchberger laisse beaucoup de liberté lors de sa mise en œuvre. Le temps d'exécution peut être réduit en appliquant diverses améliorati

- L'ordre dans lequel les paires critiques f, g sont sélectionné es influence le temps de fonctionnement.
- On peut utiliser les critères de Buchberger pour éviter des réductions inutiles à zéro.
- Des algorithmes existent [9, 4] pour convertir une base de Gröbner (d'un idéal de dimension zéro) dans un ordre monôme en une base de Gröbner dans un autre ordre monôme, ainsi nous pouvons calculer par rapport

à la ordre lexicographique inversé de degré d'abord, puis convertissez le résultat en ordre lexicographique.
Buchberger lui-même a donné deux critères pour éviter des réductions inutiles à zéro.

Définition 27 (Premier critère de Buchberger [2]). *Soit $f, g \in \mathbb{F}$ avec les principaux termes disjoints, c'est- à-dire $GCD(LM(f), LM(g)) = 1$. Alors $S(f,g) \underset{\{f,g\}}{\longrightarrow} 0$.*

Proof. Voir [2, p.222].

Définition 28 (Deuxième critère de Buchberger [2]). *Soit F un sous-ensemble fini de $\mathbb{F}$ et $g_0, p, g_1 \in \mathbb{F}$ tel que ce qui suit tient :*

1. *$LM(p) \mid LCM(LM(g_0), LM(g_1))$, et*
2. *$S(g_i, p)$ a une t_i-représentation w.r.t. F avec $t_i < LCM(LM(g_i), LM(p))$ $fori = 0, 1$.*

alors $S(g_0, g_1) \underset{F}{\longrightarrow} 0$.

Proof. Voir [2, p.224ff].

L'instanciation standard de ces deux critères est l'installation Gebauer-Möller [10] qui est implémentée dans la plupart des systèmes d'algèbre informatique qui implémentent les algorithmes de base de Gröbner. Plus loin dans ce texte, nous discuterons d'autres algorithmes améliorés de base de Gröbner tels que F_4 et F_5 , tous deux dus à Jean-Charles Faugère.

Nous considérerons également plus tard le calcul des bases de Gröbner jusqu'à un certain degré D. C'est-à-dire que nous exécutons, par exemple, l'algorithme de Buchberger mais rejetons tous les S-polynômes avec un degré $> D$. Si tous les polynômes d'entrée sont homogènes, les bases de Gröbner jusqu'à un degré D sont bien définies. Cependant, dans le cas affine, ce n'est pas vrai car le degré lors d'une réduction polynomiale peut chuter. Ainsi, le calcul jusqu'à un certain degré D dans le cas affine n'est guère plus qu'une interruption aléatoire de l'algorithme de base de Gröbner.

2.5 Résolution de systèmes polynomiaux avec les bases de Gröbner

Cette section vise à expliquer la relation entre la résolution de systèmes d'équations polynomiales et les bases de Gröbner. Nous devons d'abord définir formellement le concept de solution à un système de polynômes.

Définition 29. *Étant donné un corps $\mathbb{F}$ et un entier positif n, nous définissons l'espace affine à n dimensions sur $\mathbb{F}$ pour être l'ensemble*

$\mathbb{F}^n = \{(a_0, \ldots, a_{n-1}) : a_0, \ldots, a_{n-1} \in \mathbb{F}\}$.

Évaluation d'un polynôme $f \in \mathbb{F}$ en $(a_0, \ldots, a_{n-1}) \in \mathbb{K}^n$, où $\mathbb{K}$ est une extension algébrique de $\mathbb{F}$, est une fonction

$$f : \mathbb{K}^n \longrightarrow \mathbb{K},$$

où chaque x_i est remplacé par $a_i \in \mathbb{K}$ pour $0 \leq i < n$.

L'ensemble de toutes les solutions dans $\mathbb{K}^n$ à un système d'équations

$$f_0(x_0, \ldots, x_{n-1}) = 0, \ldots, f_{m-1}(x_0, \ldots, x_{n-1}) = 0$$

est appelée une *affine* $\mathbb{F}$-variété, formellement définie comme suit

Définition 30. *Soit $\mathbb{F}$ être un corps, $\mathbb{K}$ une extension algébrique de $\mathbb{F}$ et $f_0, \ldots, f_{m-1}$ sont des polynômes dans $\mathbb{F}[x_0, \ldots, x_{n-1}]$, c'est-à-dire que tous les coefficients sont dans $\mathbb{F}$. Nous définissons*

$V(f_0, \ldots, f_{m-1}) = \{(a_0, \ldots, a_{n-1}) \in \mathbb{K}^n : f_i(a_0, \ldots, a_{n-1}) = 0 forall 0 \leq i < m\}$.

On appelle $V(f_0, \ldots, f_{m-1})$ l'affine $\mathbb{F}$-vari été définie par $f_0, \ldots, f_{m-1}$.

Notez que$\mathbb{F}$ dans "affine $\mathbb{F}$ -variété "se réfère au corps des coefficients et non à la solution.définie

Définition 31 (PoSSo)**.** *Étant donné un ensemble fini $F = \{f_0, \ldots, f_{m-1}\} \subset \mathbb{F}[x_0, \ldots, x_{n-1}]$ des polynômes multivarié s dans P nous appelons PoSSo le problème de la recherche de la variété affine de F.*

Lemme 32. *si $f_0, \ldots, f_{s-1}$ et $g_0, \ldots, g_{t-1}$ sont des bases du même idéal dans P, de sorte que*

$$\langle f_0, \ldots, f_{s-1} \rangle = \langle g_0, \ldots, g_{t-1} \rangle,$$

alors

$$V(f_0, \ldots, f_{s-1}) = V(g_0, \ldots, g_{t-1}).$$

Démonstration. Chaque $f \in \langle f_0, \ldots, f_{s-1} \rangle$ est également dans $\langle g_0, \ldots, g_{t-1} \rangle$ et peut donc ˆ**e**tre exprimé comme

$$f = h_0 g_0 + \cdots + h_{t-1} g_{t-1}.$$

Par conséquent, chaque $a = (a_0, \ldots, a_{n-1}) \in V(g_0, \ldots, g_{t-1})$ satisfait $f(a) = 0$ et vice versa pour tout $g \in \langle g_0, \ldots, g_{t-1} \rangle$. Cela montre que les deux variétés sont constituées des mˆ**e**mes points.

□

Définition 33. *Soit $I \subset \mathbb{F}$ un idéal. On définit $V(I)$ comme l'ensemble*

$$\{(a_0, \ldots, a_{n-1}) \in \mathbb{K}^n : f(a_0, \ldots, a_{n-1}) = 0 \ forall f \in I\}$$

pour une extension algébrique $\mathbb{K}$ de $\mathbb{F}$.

Une conséquence du théorème de base de Hilbert (Théorème 10) est que la variété correspondant à un ensemble de polynômes F est égale à la variété de l'idéal enjambé par cet ensemble de polynômes

Proposition 34. *$V(I)$ est une variété affine. En particulier, si $I = \langle f_0, \ldots, f_{m-1} \rangle$, alors $V(I) = V(f_0, \ldots, f_{m-1})$.*

Proof. Voir [5, p.77]

Ainsi, une instance du problème PoSSo peut être considérée comme la base d'un idéal I.S'il y avait une base pour le même idéal où la solution - la variété $V(I)$ - pouvait être lue directement, le problème PoSSo était résolu. Il s'avère que les bases de Gröbner satisfont à cette exigence sous certaines conditio

Pour le montrer, une notation supplémentaire doit d'abord être établie. Étant donné un I idéal dans un anneau polynomial $P = \mathbb{F}[x_0, \ldots, x_{n-1}]$ sur un corps $\mathbb{F}$ et un nombre $j \in \{0, \ldots, n-1\}$, considérons l'ensemble de tous les polynômes de I qui n'impliquent que les variables $x_{0+j}, \ldots, x_{n-1}$. Cet ensemble $I \cap \mathbb{F}[x_{0+j}, \ldots, x_{n-1}]$ est un idéal dans $\mathbb{F}[x_{0+j}, \ldots, x_{n-1}]$.

Définition 35 (Idéal d'élimination)**.** *Étant donné $I = \langle f_0, \ldots, f_{m-1} \rangle \subset \mathbb{F}[x_0, \ldots, x_{n-1}]$, le l-ème idéal d'élimination I_l est l'idéal de $\mathbb{F}[x_{0+l}, \ldots, x_{n-1}]$ défini par*

$$I_l = I \cap \mathbb{F}[x_{0+l}, \ldots, x_{n-1}].$$

Il s'avère important que le système d'équations décrit un ensemble fini de solutions. L'idéal couvert par les polynômes correspondants d'un tel système sera appelé zéro dimension. La proposition suivante fournit un critère algorithmique de finitude.

Lemme 36 (Critère de finitude)**.** *Soit $P = \mathbb{F}[x_0, \ldots, x_{n-1}]$. Pour un système d'équations correspondant à un idéal $I = \langle f_0, \ldots, f_{m-1} \rangle$, les conditions suivantes sont équivalentes.*

1. *Le système d'équations n'a qu'un nombre fini de solutions dans la clôture algébrique de $\mathbb{F}$.*
2. *Pour $i = 0, \ldots, n-1$, on a $I \cap \mathbb{F}[x_i] \neq 0$.*
3. *L'ensemble des monômes $M(P) \setminus \{LM(f) : f \in I\}$ est fini*

4. Le $\mathbb{F}$-espace vectoriel P/I est de dimension finie.

Proof. Voir [12, p.243ff].

Notez que l'algorithme de Buchberger est capable de tester la condition 3 de ce lemme.

Exemple 37. *Considérez $P = \mathbb{F}[x, y, z]$. On peut montrer que l'idéal*

$$I = \langle x + y + z, xy + xz + yz, xyz - 1\rangle$$

est de dimension zéro. Alors que dans $P' = \mathbb{F}[w, x, y, z]$ l'idéal $J = \langle x + y + z, xy + xz + yz, xyz - 1\rangle$ n'est pas de dimension nulle car il existe un ensemble infini de monômes w^i avec $i \in \mathbb{Z}$ qui n'est pas dans $LM(J)$

Si nous considérons des corps finis et ajoutons les polynômes de
à un système de polynômes, l'idéal couvert par cet ensemble combiné de polynômes est de dimension nulle car dans ce cas la condition 2 est satisfaite. Ces polynômes de corps sont définis comme suit :

Définition 38. *Soit $\mathbb{F}$ un corps d'ordre $q = p^n$, p premier et $n > 0$ Ensuite, les polynômes de corps de l'anneau $\mathbb{F}$ sont définis comme l'ensemble**

$$\{x_0^q - x_0, \ldots, x_{n-1}^q - x_{n-1}\}.$$

L'idéal couvert par cet ensemble

$$\langle\ x_0^q - x_0, \ldots, x_{n-1}^q - x_{n-1}\rangle$$

s'appelle l'idéal de corps de $\mathbb{F}$.

Corollaire 39. *Soit I un idéal dans $\mathbb{F}$. L'idéal englobé par les générateurs de I et les générateurs de l'idéal de corps a la même variété sur $\mathbb{F}$ que l'id éal I mais exclut toutes les coordonnées de $\overline{\mathbb{F}}^n \setminus \mathbb{F}^n$„ où $\overline{\mathbb{F}}$ est la fermeture algébrique $\mathbb{F}$.*

Démonstration. Tout corps fini $\mathbb{F}$ d'ordre q satisfait $x^q = x, \forall x \in \mathbb{F}$. Ainsi les équations $x_i^q - x_i = 0 : 0 \leq i < n$ sont satisfait pour chaque coordonnée possible dans $\mathbb{F}^n$ et en particulier pour chaque élément de $V(I)$. De plus, $x_i^q - x_i$ se factorise complètement sur $\mathbb{F}$ et donc au-cun point dans $\overline{\mathbb{F}}^n \setminus \mathbb{F}^n$ ne le satisfait.

□

Pour plus d'informations sur les polynômes possibles se produisant dans l'idéal décrit par un ensemble de polynômes, le Nullstellensatz de Hilbert est d'une grande importance. Il déclare qu'un polynôme sur un corps algébriquement clos ayant des zéros communs avec les polynômes dans $F = \{f_0, \ldots, f_{m-1},$ se produit à une certaine puissance dans l'idéal enjambé par F. Mais d'abord, nous devons définir l'idéal d'une variété affine $\mathbb{F}$-variété.

Définition 40. *Soit $V \subset \mathbb{F}^n$ un affine $\mathbb{F}$-variété. Ensuite, nous définissons $I(V)$ comme suit :*

$$I(V) = \{f \in \mathbb{F} \ : f(a_0, \ldots, a_{n-1}) = 0 \ forall(a_0, \ldots, a_{n-1}) \in V\}.$$

Lemme 41. *$I(V)$ est un idéal.*

Proof. Voir [5, p.31ff].

Théorème 42 (Nullstellensatz de Hilbert)**.** *Soit $\mathbb{F}$ un corps algébriquement clos. si f et $f_0, \ldots, f_{m-1} \in \mathbb{F}$ sont tels que $f \in I(V(f_0, \ldots, f_{m-1}))$, alors il existe un entier $e \geq 1$ tel que*

$$f^e \in \langle f_0, \ldots, f_{m-1} \rangle$$

$f^e \in \langle f_0, \ldots, f_{m-1} \rangle$ et inversement.

Proof. Voir [5, p.171].

L'ensemble des polynômes satisfaisant cette condition est appelé le radical de l'idéal I.

Définition 43. *Soit $I \subset P$ un idéal. Le radical de I noté $\sqrt{I}$, est l'ensemble*

$$\{f : f^e \in I \text{ pour un entier } e \geq 1\}.$$

Lemme 44. *$\sqrt{I}$ est un idéal.*

Proof. Voir [5, p.174].

Ainsi, le Nullstellensatz de Hilbert dit que $I(V(I)) = \sqrt{I}$.

Proposition 45 (Lemme de Seidenberg)**.** *Soit $\mathbb{F}$ un corps, soit $P = \mathbb{F}[x_0, \ldots, x_{n-1}]$, et soit $I \subset P$ un idéal de dimension zéro. Supposons que pour chaque $i \in \{0, \ldots, n-1\}$ il existe un polynôme non nul $g_i \in I \cap \mathbb{F}[x_i]$ tel que le plus grand diviseur commun (GCD) de g et de sa dérivée soit égal à I. Alors I est un idéal radical.*

Proof. Voir [12, p.250ff]

Considérons un ensemble d'équations polynomiales sur $\mathbb{F}_q$, pour q la puissance d'un p premier avec des solutions dans $\mathbb{F}_q^n$. Supposer

$$F = \{f_0, \ldots, f_{m-1}\} \subset \mathbb{F}_q[x_0, \ldots, x_{n-1}]$$

et les équations

$$\begin{aligned} 0 &= f_0(x_0, \ldots, x_{n-1}), \\ 0 &= f_1(x_0, \ldots, x_{n-1}), \\ &\vdots \\ 0 &= f_{m-1}(x_0, \ldots, x_{n-1}), \end{aligned}$$

telles que les solutions possibles existant dans $\overline{\mathbb{F}}^n \setminus \mathbb{F}^n$ ne nous intéresse pas. Par conséquent, il découle du lemme de Seidenberg, que l'ajout de l'ensemble

$$\{x_i^q - x_i : 0 \leq i < n\} i < n\}$$

to F, creates a radical ideal J with variety $V(J) = V(I) \bigcap \mathbb{F}^n$.

Le théorème suivant stipule qu'une base lexicographique de Gröbner G pour l'idéal radical

de dimension zéro enjambée par les polynômes du problème PoSSo et les générateurs de l'idéal de corps permet de lire la solution du problème PoSSo à partir de G.

Théorème 46 (Théorème d'élimination)**.** *Soit $\subset \mathbb{F}$ un idéal et soit G une base de Gröbner de I par rapport à l'ordre des monômes lexicographiques où $x_0 > x_1 > \cdots > x_{n-1}$. Alors pour tout $0 \leq l < n$, l'ensemble*

$$G_l = G \cap \mathbb{F}[x_{0+l}, \ldots, x_{n-1}]$$

$G_l = G \cap \mathbb{F}[x_{0+l}, \ldots, x_{n-1}]$est une base de Gröbner pour le l-ème idéal d'élimination I_l.

En d'autres termes, la base de Gröbner G a une forme triangulaire. Pour illustrer cela, considérons l'exemple suivant.

Exemple 47. *Soit $\mathbb{F} = \mathbb{F}_{127}$, $P = \mathbb{F}_{127}[x, y, z]$, le monôme ordonnant lex et considérons l'idéal*

$$I = \langle x + y + z, xy + xz + yz, xyz - 1 \rangle$$

qui s'appelle Cyclic-3*. Nous ajoutons les polynômes de corps et calculons la base de Gröbner réduite :*

$$x + y + z, y^2 + yz + z^2, z^3 - 1,$$

qui a une forme triangulaire comme prédit par le théorème d'élimination. Ce résultat peut être calculé en utilisant Sage comme suit :

```
sage: P.<x,y,z> = PolynomialRing(GF(127),order='lex')
sage: I = sage.rings.ideal.Cyclic(P)
sage: I
Ideal (x + y + z, x*y + x*z + y*z, x*y*z - 1) of \
Multivariate Polynomial Ring in x, y, z over \
Finite Field of size 127
sage: J = I + sage.rings.ideal.FieldIdeal(P)
sage: g0,g1,g2 = J.groebner_basis(); g0,g1,g2
(x + y + z, y^2 + y*z + z^2, z^3 - 1)
sage: factor(g2)
(z - 19) * (z - 1) * (z + 20)
sage: factor(g1(x,y,19))
(y - 1) * (y + 20)
sage: factor(g0(x,1,19))
x + 20
sage: all(f(107,1,19)==0 for f in I.gens())
True
sage: J.variety()
[{y: 19, z: 1, x: 107}, {y: 107, z: 1, x: 19},
{y: 1, z: 19, x: 107}, {y: 107, z: 19, x: 1},
{y: 1, z: 107, x: 19}, {y: 19, z: 107, x: 1}]
```

Ainsi, nous pouvons utiliser les bases de Gröbner pour résoudre le problème PoSSo.

2.6 Bases de Gröbner dans les anneaux de quotient

Dans cette section, nous considérons les bases de Gröbner dans les anneaux de quotient d'anneaux polynomiaux. La raison pour laquelle nous nous intéressons à ces objets est qu'il existe des implémentations efficaces d'algorithmes basés sur Gröbner dans l'anneau. $\mathbb{F}_2[x_0,\ldots,x_{n-1}]/\langle x_0^2-x_0,\ldots,x_{n-1}^2-x_{n-1}\rangle$ tel que PolyBoRi.

Définition 48. *Soit $I \subset P$ un idéal, et soit $f,g \in P$. On dit que f et g sont congruents modulo I, en écrit*

$f \equiv g \ \% \ I$,*si* $f-g \in I$.

Proposition 49. *Soit $I \subset P$ un idéal. La congruence modulo I est une relation d' équivalence sur P. Une relation d'équivalence sur un ensemble S partitionne cet ensemble en une collection de sous-ensembles disjoints appel és classes d'équivalence. Pour tout $f \in P$, la classe de f est l'ensemble*

$$[f] = \{g \in P : g \equiv f \ \% \ I\}$$

Proof. Voir [5, p.219].

Définition 50. *Le quotient de $\mathbb{F}$ module I, écrit $\mathbb{F} \ /I$, est l'ensemble des classes d'équivalence pour le module de congruen I :*

$$\mathbb{F}[x_0, \ldots, x_{n-1}]/I = \{[f] : f \in \mathbb{F}[x_0, \ldots, x_{n-1}]\}.$$

En $P = \mathbb{F} \ /I$ l'addition et la multiplication peuvent être définies comme suit :

$$\begin{aligned} [f] + [g] &= [f + g] \\ [f] \cdot [g] &= [f \cdot g]. \end{aligned} \tag{2.1}$$

Ces définitions sont indépendantes du choix du représentant de $[f]$ et $[g]$: f, g.

Théorème 51. *[5, p.221] Soit I un idéal dans $\mathbb{F}[x_0, \ldots, x_{n-1}]$. Le quotient*

$$\mathbb{F} \ /I$$

est un anneau commutatif sous les opérations somme et produit donné es dans (2.1).

Par conséquent, $Q = P/I = \mathbb{F} \ /I$ peut être appelé un anneau de quotient. $P = \mathbb{F}$ est appelé son anneau de couverture et I son idéal définissant.

Comme Q est un anneau commutatif, les idéaux peuvent y être construits avec les propriétés habituelles des idéaux. Ces idéaux sont étroitement liés aux idéaux de la bague de recouvrement P.

Théorème 52. *[5, p.223] Soit I un idéal dans $\mathbb{F}$. Les idéaux de l'anneau quotient $\mathbb{F} \ /I$ sont en correspondance un à un avec les idéaux de $\mathbb{F}$ contenant I (que est, les idéaux J satisfaisant $I \subset J \subset P$).*

Voir [5, p.223].

En particulier, nous pouvons identifier

$$I = \langle f_0, \ldots, f_{m-1}, x_0^2 - x_0, \ldots, x_{n-1}^2 - x_{n-1} \rangle \in \mathbb{F}_q[x_0, \ldots, x_{n-1}]$$

avec

$$J = \langle [f]_0, \ldots, [f]_{m-1} \rangle \in \mathbb{F}_q[x_0, \ldots, x_{n-1}]/\langle x_0^q - x_0, \ldots, x_{n-1}^q - x_{n-1} \rangle.$$

2.7 Algorithme F_4

Cette partie décrit l'algorithme F_4 dû à Jean-Charles Faugère. Premièrement, l'idée de base est donnée ; puis, l'algorithme F_4 original est présenté et discuté ; cette partie se termine par une présentation de F_4 proprement dit qui a ajouté les critères de Buchberger. F_4 a été décrit pour la première fois par son auteur dans l'article "A new efficient algorithm for computing Gröbner bases (F_4)" [7], où il introduit une nouvelle stratégie de réduction pour les algorithmes basés sur Gröbner. Cette stratégie de réduction est basée sur la liaison des bases de Gröbner à l'algèbre linéaire [13] et permet de réduire plusieurs S-polynômes à la fois au lieu d'un par un.

2.8 Matrices de coefficients et division polynomiale

La plupart des algorithmes considérés dans cette section et les suivants construisent des matrices de coefficients à partir de tuples de polynômes. Chaque tuple ordonné de polynômes $F = [f_0, \ldots, f_{m-1}]$ dans $P = \mathbb{F}$ peut être représenté par la paire A_F, v_F comme suit : Fixez un ordre des monômes sur les monômes en P et soit

$$v_F = (m_{|M(F)|-1}, \ldots, m_0)^T$$

être le vecteur contenant les monômes apparaissant dans F par ordre décroissant (y compris 1 si applicable). Soit $a_{ij} = A_F[i,j]$ le coefficient de m_j dans f_i (éventuellement zéro). Alors F peut être récupéré à partir des lignes de $A_F \cdot v_F$.

On appelle A_F la matrice de coefficients de F and $v_F$le vecteur monôme de F.

Donc par exemple,

$$\begin{aligned} f_0 =& 81z^2 + 51x + 125z + 38 \\ f_1 =& 76xy + 80y^2 + 49xz + 62yz + 45z^2 \\ f_2 =& 122x^2 + 106yz + 78z^2 + 48x + 112y \end{aligned}$$

dans $\mathbb{F}_{127}[x, y, z]$ avec un *degrevlex* d'ordre monomial peut être exprimé comme :

$$\begin{pmatrix} f_0 \\ f_1 \\ f_2 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 81 & 51 & 0 & 125 & 38 \\ 0 & 76 & 80 & 49 & 62 & 45 & 0 & 0 & 0 & 0 \\ 122 & 0 & 0 & 0 & 106 & 78 & 48 & 112 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} x^2 \\ xy \\ y^2 \\ xz \\ yz \\ z^2 \\ x \\ y \\ z \\ 1 \end{pmatrix}.$$

Le même calcul avec Sage :

```
sage: k = GF(127)
sage: P.<x,y,z> = PolynomialRing(k, order='degrevlex')
sage: f = -46*z^2 + 51*x - 2*z + 38
sage: g = -51*x*y - 47*y^2 + 49*x*z + 62*y*z + 45*z^2
sage: h = -5*x^2 - 21*y*z - 49*z^2 + 48*x - 15*y
sage: F = mq.MPolynomialSystem([f,g,h])
sage: A,v = F.coefficient_matrix()
sage: A
[  0   0   0   0   0  81  51   0 125  38]
[  0  76  80  49  62  45   0   0   0   0]
[122   0   0   0 106  78  48 112   0   0]
sage: v
[x^2]
[x*y]
[y^2]
[x*z]
[y*z]
[z^2]
[  x]
[  y]
[  z]
[  1]
```

Afin de trouver la base réduite du système linéaire de polynômes, la méthode simple consiste à noter la matrice de coefficients comme ci-dessus et à effectuer une élimination gaussienne. Considérons par exemple un système linéaire d'équations sur $\mathbb{F}_{127}[x,y,z]$: $26y+52z+62, 54y+119z+55$ and $41x+91z+13$. La matrice des coefficients est :

$$\begin{pmatrix} 0 & 26 & 52 & 62 \\ 0 & 54 & 119 & 55 \\ 41 & 0 & 91 & 13 \end{pmatrix}$$

et sa forme en échelon de rang réduit :

$$\begin{pmatrix} 1 & 0 & 0 & 29 \\ 0 & 1 & 0 & 38 \\ 0 & 0 & 1 & 75 \end{pmatrix}$$

qui correspond à $x+29, y+38$ et $z+75$.

Ceci motive la définition de l'élimination gaussienne sur un système de polynômes comme élimination gaussienne sur sa matrice de coefficients :

Considérons maintenant deux polynômes dans $\mathbb{F}_{127}[x,y,z]$ avec le degré d'ordre monôme lexicographique inverse : $f = x^2 + 2xy - 2y^2 + 14z^2 + 22z$ et $g = 3x^2 + y^2 + z^2 + x + 2z$. La matrice de coefficients correspondante est

$$\begin{pmatrix} 1 & 2 & 125 & 14 & 0 & 22 \\ 3 & 0 & 1 & 1 & 1 & 2 \end{pmatrix}$$

Input : F – a polynomial system of equations
Result : a polynomial system of equations
begin
  $A_F, v_F \longleftarrow$ coefficient matrix for F;
  $E \longleftarrow$ row echelon form of A_F;
  return rows of $E * v_F$;
end

Algorithmus 3 : Gaussian Elimination

et sa forme en échelonné réduite est

$$\begin{pmatrix} 1 & 0 & 85 & 85 & 85 & 43 \\ 0 & 1 & 20 & 28 & 21 & 53 \end{pmatrix}$$

qui correspond à

$$f'^2 + 85y^2 + 85z^2 + 85x + 43z,$$
$$g'^2 + 28z^2 + 21x + 53z.$$

Comparez ce résultat avec le reste de la division polynomiale $f/g = r = 2xy + 40y^2 + 56z^2 + 42x + 106z$ et notez que ce résultat est le même que $2g'$. Ainsi, nous pouvons utiliser l'algèbre linéaire pour effectuer une division polynomiale dans certaines situations. Cependant, cette approche simple échoue en général, comme le montre l'exemple suivant :

$$f = x^2 - 2xy - 2y^2 + 14z^2,$$
$$g = x + 2z.$$

Dans cet exemple, la forme d'échelon de ligne réduite ne diffère pas de la matrice de coefficients initiale et ne parvient donc pas à fournir une réduction polynomiale puisque x n'est pas un monôme de f. D'autre part, x divise deux monômes de f, à savoir x^2 et xy et divise ainsi le monôme de tête de f. Pour effectuer une réduction polynomiale, nous pouvons inclure tous les multiples $m \cdot g$ de g tels que$LM(m \cdot g) = m \cdot LM(g) \in M(f)$. Cela donne un système de quatre polynômes :

$$f = x^2 - 2xy - 2y^2 + 14z^2,$$
$$x \cdot g = x^2 + 2xz,$$
$$y \cdot g = xy + 2yz,$$
$$g = x + 2z,$$

dont la matrice de coefficients est

$$\begin{pmatrix} 1 & -2 & -2 & 0 & 0 & 14 & 0 & 0 \\ 1 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 \end{pmatrix}.$$

.La forme d'échelon de ligne réduite est

$$\begin{pmatrix} 1 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 125 & 120 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 \end{pmatrix}$$

ce qui correspond à :

$$\begin{aligned} f'^2 + 2xz, \\ g' = xy + 2yz, \\ g''^2 + xz + 125yz + 120z^2, \\ g = x + 2z. \end{aligned}$$

Encore une fois, comparez avec le reste de la division polynomiale $r = f/g = -2y^2 + 4yz + 18z^2$ et notez que le terme principal de r correspond au terme principal de g'' '. Nous reviendrons plus tard sur le fait que g'' contient le monôme xz mais que le reste de f/g n'en contient pas. Pour l'instant, nous soulignons que c'est l'idée centrale de l'algorithme F_4.

2.9 L'Original F_4

Étant donné un tuple F ordonné fini de polynômes linéaires dans P, nous appelons la base de Gröbner (réduite) de ces polynômes $\tilde{F}$.Une matrice de coefficients $\tilde{A}$ peut être construite pour $\tilde{F}$. Cette matrice $\tilde{A}$ est exactement la forme d'échelon de ligne (réduite) de A_F et $\tilde{F}$ est appelé la base d'échelon de ligne de F..

De même, $A = A_F$ peut être construit pour n'importe quel tuple de polynômes contenant des polynômes linéaires et non linéaires et la forme d'échelon de ligne (réduite) pour A - appelée $\tilde{A}$ - peut être calculée. Alors $\tilde{F}$ construite à partir de $\tilde{A}$ est appelée la forme d'échelon de ligne de F.Une propriété intéressante des formes d'échelon de ligne de F est la suivante :

Soit $\tilde{F}^+$ l'ensemble

$$\{g \in \tilde{F} : LM(g) \notin LM(F)\}.$$

Si les éléments de $\tilde{F}^+$ sont joints à un sous-ensemble H du F original, tel que

$$LM(H) = LM(F) \text{ and } |H| = |LM(F)|$$

est vrai, alors l'idéal $\langle F \rangle$ est couvert par $H \cup \tilde{F}^+$. Officiellement :

Théorème 53. *[7, p.4] Soit $\mathbb{F}$ un corps et F un tuple fini d'éléments dans l'anneau polynomial $P = \mathbb{F}[x_0, \ldots, x_{n-1}]$. Soit A la matrice de coefficients de F et $\tilde{A}$ la forme échelonnée de ligne de cette matrice. Enfin, soit $\tilde{F}$ le tuple fini de polynômes correspondant à $\tilde{A}$.*

Pour tout sous-ensemble $H \subseteq F$ tel que $LM(H) = LM(F)$ et $|H| = |LM(F)|$, $G = \tilde{F}^+ \cup H$ est une base triangulaire de l'espace de $\mathbb{F}$-combinaisons linéaires de F. Soit pour tout $f = \sum_{i=0}^{m-1} c_i f_i$ avec $c_i \in \mathbb{F}$ il existe des éléments λ_k de $\mathbb{F}$ et g_k éléments de G tels que $f = \sum_k \lambda_k g_k$, $LM(g_0) = LM(f)$ et $LM(g_k) > LM(g_{k+1})$.

Au lieu de calculer la réduction de chaque S-polynôme individuellement, F_4 crée une sélection de paires critiques $p_{ij} = (f_i, f_j)$, pour f_i, f_j dans la base intermédiaire G' et passe les deux paires

$$(\sigma_{i,j}, f_i), (\sigma_{j,i}, f_j)$$

avec $\sigma_{i,j} = LCM(LM(f_i),\ LM(f_j))/LM(f_i)$ à la fonction de réduction. Notez que pour chaque paire critique les tuples $(\sigma_{i,j}, f_i)$ et $(\sigma_{j,i}, f_j)$ correspondent au produit non évalué pour chaque composante de $S(f_i, f_j)$. Cette paire est construite dans une routine appelée Pair. La stratégie de sé lection recommandée dans [7] est la stratégie de sélection normale :

Définition 54 (Stratégie normale)**.** *Soit $\mathcal{P}$ un tuple de paires critiques et soit $LCM(p_{ij})$ le plus petit commun multiple des monômes principaux des deux parties de la paire critique $p_{ij} = (f_i, f_j)$. De plus, soit $d = \min\{\deg(LCM(p)), p \in \mathcal{P}\}$, désigne le degré minimum de ces multiples les moins communs de p dans $\mathcal{P}$.. Alors la stratégie de sélection normale sélectionne le sous-ensemble $\mathcal{P}'$ de $\mathcal{P}$ avec $\mathcal{P}' = \{p \in \mathcal{P} \mid \deg(LCM(p)) = d\}$.*

Définition 55. *Soit p_{ij} une paire critique f_i, f_j comme ci-dessus.*

- *$Left(p_{ij})$ désigne la paire $(\sigma_{i,j}, f_i) \in M \times P$ où $\sigma_{i,j} = LCM(p_{ij})/LM(f_i)$ et*
- *$Right(p_{ij})$ désigne la paire $(\sigma_{j,i}, f_j) \in M \times P$ où $\sigma_{j,i} = LCM(p_{ij})/LM(f_j)$.*

Ces définitions sont étendues à des ensembles de paires critiques en les appliquant individuellement à leurs membres. $\mathcal{L}_d$ désigne Gauche $Left(\mathcal{P}_d) \cup Right(\mathcal{P}_d)$.

Exemple 56. *À titre d'exemple, considérons*

$$f_0 = -45xy + 36y^2 - 18xz - 63z^2 + 17,$$
$$f_1 = -34y^2 - 53xz - 52yz - 58z^2 - 47x$$

dans l'anneau $\mathbb{F}_{127}[x, y, z]$ avec l'ordre monôme lexicographique inversé. Puis $Left(p_{0,1}) = (y, f_0)$ et $Right(p_{0,1}) = (x, f_1)$, puisque $LCM(LM(f_0), LM(f_1)) = xy^2$.

Maintenant que les paires critiques à réduire sont sélectionnées, des réducteurs doivent être ajoutés à la base intermédiaire G' pour réduire ces paires, tout comme dans l'exemple de la section 2.8. L'ajout de réducteurs se fait par une routine appelée SYMBOLIC PREPROCESSING$_o$ qui agit sur $Left(\mathcal{P}_d) \cup Right(\mathcal{P}_d)$.

Définition 57 (Réducteur)**.** *Lors de l'exécution d'un algorithme pour calculer les bases de Gröbner, nous appelons un polynôme r satisfaisant*

$$LM(r) \in M(F) \setminus LM(F).$$

un Réducteur.

Notez que les termes principaux de $Left(\mathcal{P}_d) \cup Right(\mathcal{P}_d)$ n'ont pas besoin d'un réducteur ajouté au système car ils correspondent aux deux composantes d'un S-polynôme qui n'ont pas encore été réduites, donc un composant annulera le terme principal de l'autre.

```
Input : L – a finite subset of M × P
Input : G – a finite subset of P
Result : a finite subsef of P
begin
    F ⟵ {t · f, ∀(t, f) ∈ L};
    Done ⟵ LM(F);
    while M(F) ≠ Done do
        m ⟵ an element in M(F) \ Done;
        add m to Done;
        if ∃ g ∈ G : LM(g) | m then
            u = m/LM(g);
            add u · g to F;
        end
    end
    return F;
end
```

Algorithmus 4 : SYMBOLIC PREPROCESSING$_o$

SYMBOLIC PREPROCESSING$_o$ fait plus de travail que dans l'exemple de la section 2.8. Il continuera d'ajouter de nouveaux réducteurs tant qu'aucun monôme dans l'ensemble intermédiaire F n'est pris en compte. Cette différence explique pourquoi g'' n'a pas été complètement réduit dans l'exemple : $z \cdot g$ n'a pas été ajouté pour tenir compte du monôme yz nouvellement introduit. SYMBOLIC PREPROCESSING$_o$, quant à lui, garantit une réduction complète en ajoutant de nouveaux réducteurs jusqu'à ce que chaque monôme se produisant dans le système soit pris en compte. Notez que SYMBOLIC PREPROCESSING$_o$ n'ajoute à M que des monômes plus petits que $LM(F)$ et qu'il n'y a qu'une infinité de ces monômes. Le SYMBOLIC PREPROCESSING$_o$, est utilisé par une fonction appelée

REDUCTION$_o$ qui réduit simultanément les polynômes correspondant à plusieurs paires critiques.

Input : L – a finite subset of $M \times P$
Input : G – a finite subset of P
Result : a finite subsef of P
begin
  $F \longleftarrow$ SYMBOLIC PREPROCESSING$_o$(L, G);
  $\tilde{F} \longleftarrow$ GAUSSIAN ELIMINATION(F);
  $\tilde{F}^+ \longleftarrow \{f \in \tilde{F} \mid \mathrm{LM}(f) \notin \mathrm{LM}(F)\}$;
  return $\tilde{F}^+$;
end

Algorithmus 5 : REDUCTION$_o$

Les S-polynômes qui ne se réduisent pas à zéro dans l'algorithme de Buchberger, prolongent l'idéal englobé par les principaux termes de la base intermédiaire. De cette façon, une chaîne ascendante d'idéaux principaux de termes est obtenue. De même, les principaux termes des éléments de $\tilde{F}^+$contribuent à l'idéal enjambé par les principaux termes de la base intermédiaire. Ceci est formalisé dans le lemme suivant : plusieurs paires critiques.

Lemme 58. *[16, p.59] Soit $\tilde{F}^+$ la sortie de* REDUCTION *appliquée à $\mathcal{L}_d$ par rapport à G. Pour tout $f \in \tilde{F}^+$, $LM(f)$ n'est pas un élément de $\langle LM(G)\rangle$.*

Démonstration. [16, p.59] Soit F l'ensemble calculé par l'algorithme SYMBOLIC PREPROCESSING$_o(\mathcal{L}_d, G)$. Supposons pour une contradiction que $\exists\ h \in \tilde{F}^+$ tel que $t = \mathrm{LM}(h) \in \langle \mathrm{LM}(G)\rangle$. donc $\mathrm{LM}(g)$ divise t pour quelque $g \in G$. Nous avons que t is in $M(\tilde{F}^+) \subset M(\tilde{F}) \subset M(F)$ et est réductible en haut de g, donc $\frac{t}{\mathrm{LM}(g)}g$ est inséré dans F par SYMBOLIC PREPROCESSING$_o$ (ou un autre produit avec le m^**e**me mon$\hat{\mathrm{o}}$me principal). Cela contredit le fait que nous avons besoin de $\mathrm{LM}(h) \notin \mathrm{LM}(F)$.

□

Le lemme suivant assure que les éléments ajoutés à la base intermédiaire sont des membres de l'idéal $\langle G\rangle$.

Lemme 59. *[16, p.59] Soit $\tilde{F}^+$ comme dans le lemme 58. Puis $\tilde{F}^+ \subset \langle G\rangle$.*

Démonstration. [16, p.60] Chaque $f \in \tilde{F}^+$ est une combinaison linéaire d'éléments de $\mathcal{L}_d$ et les réducteurs R, qui sont tous deux des sous-ensembles de $\langle G\rangle$.

□

Le lemme suivant indique que tous les S-polynômes de l'ensemble des combinaisons linéaires $\mathbb{F}$ possibles de $\mathcal{L}_d$ se réduisent à zéro par un sous-ensemble de $\tilde{F}^+ \cup G$. Ceci est utilisé pour prouver l'exactitude de l'algorithme par le critère énoncé dans le théorème 25

Lemme 60. *[16, p.60] Soit $\tilde{F}^+$ comme dans le lemme 58. Pour toutes les $\mathbb{F}$ -combinaisons linéaires f d'éléments de $\mathcal{L}_d$, on a que $f \underset{\tilde{F}^+ \cup G}{\longrightarrow} 0$.*

Démonstration. [16, p.60] Soit f une combinaison linéaire d'éléments de $\mathcal{L}_d$. Supposons que F soit la sortie du SYMBOLIC PREPROCESSING$_o$ de $\mathcal{L}_d$ et G. Par construction, $\mathcal{L}_d$ est un sous-ensemble de F et donc en raison de Theorem 53, ces éléments sont une

combinaison de la base triangulaire $\tilde{F}^+ \cup H$ pour un sous-ensemble approprié $H \subset F$. Les éléments de of H sont soit des éléments de $\mathcal{L}_d$ ou (par construction en SYMBOLIC PREPROCESSING$_o$) de la forme $x^\alpha g$, pour $g \in G$ et $\alpha \in \mathbb{N}^n$, et f peuvent donc s'écrire comme

$$f = \sum_i a_i f_i + \sum_j a_j x^{\alpha_j} g_j,$$

pour $f_i \in \tilde{F}^+$ et $g_j \in G$, $a_i, a_j \in \mathbb{F}$ et $\alpha_j \in \mathbb{Z}^n_{\geq 0}$. Ainsi l'algorithme de division donne un reste égal à 0 pour un tuple convenable d'\{}'{e}léments dans $\tilde{F}^+ \cup G$ et il existe donc une chaîne de réduction à 0.

□

Sur la base de ces résultats, nous pouvons formuler une première version de F_4 et prouver

son exactitude.

Théorème 61. *L'algorithme 6 calcule une base de Gröbner G pour un idéal enjambé par F,tel que $F \subseteq G$, en un nombre fini d'é tapes.*

Démonstration. [7, p.8] La résiliation et l'exactitude doivent ître prouvées :

Termination Supposons que la boucle while ne se termine pas. Il existe une suite ascendante (d_i) de nombres naturels telle que $\tilde{F}^+_{d_i} \neq \emptyset$ pour tous i. Choisissez n'importe quel $q_i \in \tilde{F}^+_{d_i}$. Soit U_i l'idéal $U_{i-1} + \langle \mathrm{LM}(q_i) \rangle$ pour $i > 1$ et $U_0 = \{0\}$. du Lemma 58 ($\mathrm{LM}(h) \notin \mathrm{LM}(G)$) il s'ensuit que $U_{i-1} \subsetneq U_i$ comme les éléments de $\tilde{F}^+_{d_i}$ sont ajoutés à G à la fin de chaque boucle. Cette chaîne infinie d'idé aux contredit le fait que P est noéthérien .

Correctness G is $\bigcup_{d \geq 0} \tilde{F}^+_d$. Nous affirmons que l'instruction suivante sont des invariants de boucle de la boucle while :

```
Input : F – a tuple of polynomials f_0, ..., f_{m-1}
Result : a Gröbner basis for F
begin
    G, d ⟵ F, 0;
    F̃_d^+ ⟵ F;
    P ⟵ {Pair(f, g) : ∀f, g ∈ G with g > f};
    while P ≠ ∅ do
        d ⟵ d + 1;
        P_d ⟵ all pairs ∈ P with minimal degree;
        P ⟵ P \ P_d;
        L_d ⟵ Left(P_d) ⋃ Right(P_d);
        F̃_d^+ ⟵ Reduction_o(L_d, G);
        for h ∈ F̃_d^+ do
            P ⟵ P ⋃ {Pair(f, h) : ∀f ∈ G};
            add h to G;
        end
    end
    return G;
end
```

Algorithmus 6 : Original F_4

— G est un sous-ensemble fini de $\mathbb{F}[x_0, \ldots, x_{n-1}]$ such that $F \subset G \subset \langle F \rangle$ et

— les S-polynîmes pour tout $g_0, g_1 \in G$, tels que $\{g_0, g_1\} \subsetneq \mathcal{P}$ se réduisent à zéro par rapport à G.

La première revendication est une conséquence immédiate du lemme Lemma 59. Pour le second, si $\{g_0, g_1\} \subsetneq \mathcal{P}$, cela signifie que $\textsc{Pair}(g_0, g_1)$ a été sélectionné à une étape précédente (say d). Ainsi $Left(\textsc{Pair}(g_0, g_1))$ et $Right(\textsc{Pair}(g_0, g_1)$ sont dans $\mathcal{L}_d$ et donc le S-polynôme de g_0, g_1 est un $\mathbb{F}$-linéair combinaison d'éléments de $\mathcal{L}_d$. Par conséquent, par le Lemma 60 il se réduit à zéro par rapport à G.

□

2.10 Amélioration de F_4

Bien que le F_4 original présente la nouvelle stratégie de réduction utilisant l'algèbre linéaire, il ne constitue pas un algorithme efficace car il considère trop de paires critiques. Dans [7], Faugère présente également une version améliorée de son algorithme qui a les critères de Buchberger "insérés". Faugère propose d'utiliser l'installation Gebauer et Möller [10].

L'algorithme principal reste quasiment inchangé, sauf qu'une fonction Update est appelée pour créer le tuple de paires critiques. Ainsi, au lieu d'ajouter toutes

les paires critiques, les seules paires ajoutées sont celles qui survivent à la routine de mise à jour, qui vérifie les premier et deuxième critères de Buchberger. Une telle routine peut être trouvée par exemple dans [2].

Input : F – a tuple of polynomials $f_0, \ldots, f_{m-1}$
Result : a Gröbner basis for F
begin
  $G, d \longleftarrow \varnothing, 0$;
  $P \longleftarrow \varnothing$;
  while $F \neq \varnothing$ **do**
    $f \longleftarrow \min\{F\}$;
    $F \longleftarrow F \setminus \{f\}$;
    $G, P \longleftarrow$ UPDATE(G, P, f);
  end
  while $P \neq \varnothing$ **do**
    $d \longleftarrow d + 1$;
    $P_d \longleftarrow$ all pairs $\in P$ with minimal degree;
    $P \longleftarrow P \setminus P_d$;
    $\mathcal{L}_d \longleftarrow \text{Left}(P_d) \bigcup \text{Right}(P_d)$;
    $\tilde{F}_d^+, F_d \longleftarrow$ REDUCTION$(\mathcal{L}_d, G, (F_k)_{k=1,\ldots,d-1})$;
    for $h \in \tilde{F}_d^+$ **do**
      $G, P \longleftarrow$ UPDATE(G, P, h);
    end
  end
  return G;
end

Algorithmus 7 : F_4

Les routines REDUCTION$_o$ et SYMBOLIC PREPROCESSING$_o$ sont adaptées comme suit et perdent leur indice o. Le principal ajout est l'introduction de la routine SIMPLIFY dans le SYMBOLIC PREPROCESSING..

La seule nouvelle routine est l'algorithme SIMPLIFY qui tente de trouver une meilleure représentation pour un élément donné $t \cdot f$. Dans le F_4 original, seules les matrices réduites $\tilde{F}^+$ ont été conservées. Dans F_4 les matrices d'entrée F_d sont stockées et utilisées pour trouver des représentations $t' \cdot f'$ pour $t \cdot f$ telles que$t' \leq t$ et $LM(t' \cdot f') = LM(t \cdot f)$. Intuitivement, l'idée derrière cette routine est de trouver un polynôme avec le terme principal $LM(t \cdot f)$ qui avait déjà plus de réductions appliquées.

Comme le sous-programme SIMPLIFY est le changement le plus visible de l'algorithme, le théorème principal de l'algorithme SIMPLIFY est énoncé et prouvé ci-dessous. Il stipule que l'on peut remplacer les éléments $t \cdot f$ par tout ce que SIMPLIFY renvoie sans perdre aucune information.

Input : L – a finite subset of $M \times P$
Input : G – a finite subset of P
Input : $\mathcal{F}$ – $(F_k)_{k=1,\ldots,d-1}$ a tuple of finite subsets of P
Result : a finite subsef of P
begin
  $F \longleftarrow \{\text{SIMPLIFY}(m, f, \mathcal{F}),\ \forall (t, f) \in L\}$;
  $Done \longleftarrow LM(F)$;
  while $M(F) \neq Done$ **do**
    $m \longleftarrow$ an element in $M(F) \setminus Done$;
    add m to $Done$;
    if $\exists\ g \in G : \mathrm{LM}(g) | m$ **then**
      $u = m/\mathrm{LM}(g)$;
      add $\text{SIMPLIFY}(u, f, \mathcal{F})$ to F;
    end
  end
  return F;
end

Algorithmus 8 : SYMBOLIC PREPROCESSING

Lemme 62. *[7, p.10] si (t', f') est le résultat de* $\text{SIMPLIFY}(t, f, \mathcal{F})$*, alors $LM(t' \cdot f') = LM(t \cdot f)$. de plus $\tilde{\mathcal{F}}^+$ désigne $(\tilde{F}_k^+)_{k=1,\ldots,d-1}$, alors il existe $0 \neq \lambda \in P$, et $r \in \langle \tilde{\mathcal{F}}^+ \cup\ \mathcal{F} \rangle$ tel que $tf = \lambda \cdot t' \cdot f' + r$ avec $LM(r) < LM(t \cdot f)$.*

Proof. Voir [7, p.10].

En utilisant ce lemme et le fait que UPDATE ne supprime que les paires qui se réduiraient à zéro de toute façon, il est facile de montrer que F_4 calcule une base de Gröbner en un nombre fini d'étapes. Le lecteur intéressé est renvoyé à [7] pour une preuve formelle.

Input : L – a finite subset of $M \times P$
Input : G – a finite subset of P
Input : $\mathcal{F} - (F_k)_{k=1,\ldots,d-1}$ a tuple of finite subsets of P
Result : a finite subsef of P
begin
  $F \longleftarrow$ Symbolic Preprocessing$(L, G, \mathcal{F})$;
  $\tilde{F} \longleftarrow$ Gaussian Elimination(F);
  $\tilde{F}^+ \longleftarrow \{f \in \tilde{F} \mid \mathrm{LM}(f) \notin \mathrm{LM}(F)\}$;
  return $\tilde{F}^+$,F;
end

Algorithmus 9 : Reduction

Input : $t - \in M$
Input : $f - \in P$
Input : $\mathcal{F} - (F_k)_{k=1,\ldots,d-1}$ a tuple of finite subsets of P
Result : an element in P
begin
  for $\{u \in M(P) \mid u|t\}$ **do**
    if $\exists j \mid 1 \leq j < d, uf \in F_j$ **then**
      $\tilde{F}_j \longleftarrow$ row echelon form of F_j;
      there exists a (unique) $p \in \tilde{F}_j$ such that $\mathrm{LM}(p) = \mathrm{LM}(uf)$;
      if $u \neq t$ **then**
        return Simplify$(t/u, p, \mathcal{F})$;
      else
        return p;
      end
    end
  end
  return tf;
end

Algorithmus 10 : Simplify

Chapitre 3

Attaques algébriques par canaux auxiliaires

3.1 Description algébrique de l'AES

3.1.1 Introduction

Après avoir passé en revue quelques notions de cryptographie dans le premier chapitre, nous axons sur les algorithmes de chiffrement symétriques qui sont rapides et bien adaptés au chiffrement de grande quantité de données.

Le protocole de chiffrement AES offre la possibilité de chiffrer l'information avec des clés de petites tailles et par conséquent avec une faible complexité calculatoire.

La plupart des attaques connues sur les algorithmes de chiffrement par blocs comme la cryptanalyse différentielle et sa version améliorée : l'attaque boomerang , la cryptanalyse linéaire , la cryptanalyse intégrale et son application sur l'algorithme Square, sont des attaques probabilistes et s'appuient sur le fait que l'attaquant dispose d'une grande quantité de paires de textes clairs/textes chiffrés.

Comme nous l'avons vu dans l'introduction de ce chapitre, les concepteurs de l'AES ont utilisé des outils algébriques pour fournir à leur algorithme, un niveau de garantie inégalé contre les techniques de cryptanalyse statistique standards. Pourtant dès 2001, une nouvelle voie de cryptanalyse de l'AES fait son apparition, ironiquement cette voie utilise l'algèbre pour attaquer Rijndael.

En effet, fin 2001, Ferguson, Schroeppel et Whiting montrent comment écrire Rijndael en une équation algébrique fermée 1 sur $GF(2^8)$. En s'appuyant sur le

fait que, dans un corps de Galois, l'élévation au carré est une opération linéaire, ils proposent une écriture de l'AES sous la forme d'une équation algébrique contenant 250 termes pour une taille de clef de 128 bits et 2 70 termes pour une clef de 256 bits. La cryptanalyse de l'AES dépend donc maintenant de la difficulté de résolution d'une telle équation dans $GF(2^8)$.

Quelques mois plus tard, début 2002, Courtois et Pieprzyk publient un article dans lequel ils décrivent une nouvelle attaque contre Rijndael et Serpent.[15]

Le document décrit une attaque que les auteurs qualifient de plus efficace que la force brute contre Serpent et peut-être aussi contre Rijndael. Leur attaque repose, dans un premier temps, sur la description de l'AES sous la forme d'un système d'équations polynomiales multivariées sur $GF(2)$ et, dans un deuxième temps, sur un algorithme de résolution de ce système d'équations baptisé Extended Sparse Linearization (XSL). Selon les auteurs, les points clefs de leur attaque sont que les équations sont quadratiques, que le système est surdéterminé et que leur représentation matricielle est creuse.

Toujours en 2002, dans le but de diminuer la complexité de la cryptanalyse de l'AES liée au fait que les opérations y sont basées dans deux corps de Galois différents, $GF(2)$ et $GF(2^8)$, Murphy et Robshaw créent le Big Encryption System (BES).[15]

Dans l'AES les blocs de textes clairs sont intégrés dans un tableau d'états qui est transformé par les différentes fonctions de chiffrement des tours. Tandis que l'AES a un tableau d'états de 16 octets, le BES travaille sur un tableau d'états de 128 octets. Ainsi, si nous notons A l'espace du tableau d'états de l'AES et B l'espace du tableau d'états du BES, nous avons les équivalences décrites dans le tableau suivant (voir fig. 3).

A	Tableau d'états de l'AES	Espace vectoriel F^{16}
B	Tableau d'états du BES	Espace vectoriel F^{128}
B_A	Sous-ensemble de B correspondant à A	Espace vectoriel F^{128}

Fig 3 : Définition et équivalences entre les tableaux d'états de l'AES et du BES

3.1.2 Équation pour la S-Box

La S-Box de l'AES est la composition de trois opérations algébriques simples :

- une inversion dans $GF(2^8)$ avec $0^{-1} \to 0$

- une fonction de permutation linéaire : chaque bit en sortie est la somme de certains bits en entrée ;

- l'addition de la constante 63

La fonction d'inversion est donnée par $x \to x^{-1} = x^{254}$

et la fonction de permutation est donnée par la fonction polynomiale

$$x \to 05x^{254}+09x^{253}+f9x^{251}+25x^{247}+f4x^{239}+01x^{223}+b5x^{191}+8fx^{127} \quad (1)$$

Ainsi, la S-Box est définie par la fonction polynomiale sur F :

$$s(x) = 63 + \sum w_d x^{255-2^d} \quad (2)$$

où les w_d sont définis dans la fonction polynomiale décrite par l'équation (1).

L'équation (2) peut être simplifiée en effectuant les approximations suivantes :

— suppression de la constante 63 et remplacement par l'ajout d'une constante spécifique à la clef de chiffrement ;

— suppression de la puissance 255 en partant du principe que $x^{255} = 1$ pour tous les x sauf quand x = 0.

L'équation (2) devient alors :

$$s(x) = \sum_{d=0}^{7} w_d x^{-2^d} \qquad = \sum_{d=0}^{7} \frac{w_d}{x^{2^d}}$$

3.1.3 Équation pour un tour de l'AES

En utilisant la notation détaillée de L'AES , $a_{i,j}^{(r)}$ correspond à l'octet situé dans la ligne i, colonne j du tableau d'états au début du tour r.

De même, $s_{i,j}^{(r)}$ représente l'octet situé à la position (i, j) après l'exécution de la fonction *subBytes* du tour r.

Si l'on détaille le fonctionnement d'un tour, nous obtenons tout d'abord, à partir de l'équation 3, après application de la S-Box à chaque octet du tableau d'états :

$$s_{i,j}^{(r)} = S(a_{i,j}^{(r)}) = \sum_{d=0}^{7} w_{d_r} (a_{i,j}^{(r)})^{-2^{d_r}}$$

Ensuite, en prenant $t_{i,j}^{(r)}$ comme étant l'octet à la position (i, j) après applicationde la fonction *ShiftRows*, le tableau d'états peut s'écrire sous la forme :

$t_{i,j}^{(r)} = S(a_{i,j+i}^{(r)}) = \sum_{d=0}^{7} w_{d_r} (a_{i,j+i}^{(r)})^{-2^{d_r}}$

Définissons maintenant $m_{i,j}^{(r)}$ comme étant l'octet à la position (i, j) après application de la fonction *MixColumn*. Cette dernière peut s'écrire de la façon suivante :

$m_{i,j}^{(r)} = \sum_{e_r,d_r}^{3} v_{i,e_r} t_{e_r,j}^{(r)}$

où les $v_{i,j}$ sont les coefficients de la matrice de diffusion :

$$M = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 03 \end{pmatrix}$$

À partir de là, après l'application de la fonction *MixColumns*, le tableau d'états s'écrit sous la forme :

$$\begin{aligned} m_{i,j}^{(r)} &= \sum_{e_r,d_r}^{3} v_{i,e_r} \cdot \sum_{d=0}^{7} w_{d_r} (a_{e_r,j-e_r}^{(r)})^{-2^{d_r}} \\ &= \sum_{e_r,d_r}^{3} \sum_{d=0}^{7} w_{i,e_r,d_r} (a_{e_r,e_r+j}^{(r)})^{-2^{d_r}} \end{aligned}$$

Enfin, la dernière étape est réalisée par la fonction *AddRoundKey* qui ajoute la clef :

$$\begin{aligned} a_{i,j}^{(r+1)} &= m_{i,j}^{(r)} + k_{i,j}^{(r)} \\ &= k_{i,j}^{(r)} + \sum_{e_r,d_r}^{3} \sum_{d_r=0}^{7} w_{i,e_r,d_r} (a_{e_r,e_r+j}^{(r)})^{-2^{d_r}} \end{aligned} \tag{4}$$

où $k_{i,j}^{(r)}$ est la clef du tour r à la position (i, j).

L'équation 4 est donc une expression algébrique décrivant les transformations du tableau d'états durant un tour.

La version la plus simple de l'AES compte 10 tours, ce qui, en généralisant l'équation 4 et en supprimant les symboles de sommations, correspond finalement à une expression algébrique comptant environs 250 termes 2. Si nous souhaitions stocker une telle expression, en partant du principe q'un terme correspond à un octet, nous aurions besoin de 1024 To d'espace de stockage.

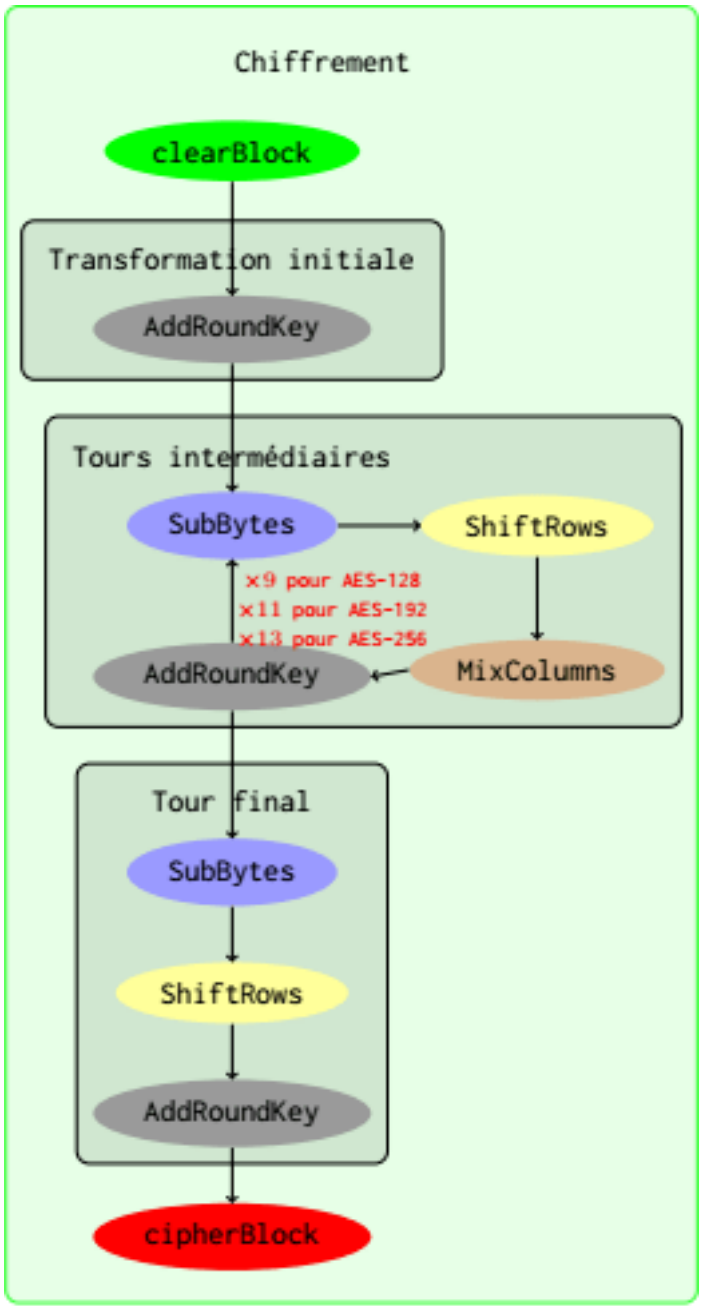


Fig 2 – Le processus de

1 : $function\ Cipher(bytein[4 * Nb], byteout[4 * Nb], wordw[Nb * (Nr + 1)])$

2 : $bytestate[4, Nb]$

3 : $state \leftarrow in$

4 : $AddRounkey(state, w[0, Nb - 1])$

5 : $for round = 1 step 1 to Nr - 1 do$

6 : $SubBytes(state)$

7 : $ShiftRows(state)$

8 : $MixColumns(state)$

9 : $AddRoundKey(state, w[round * Nb, (round + 1) * Nb - 1])$

10 : $endfor$

11 : $SubBytes(state)$

12 : $ShiftRows(state)$

13 : $AddRounkey(state, w[Nr * Nb, (Nr + 1) * Nb - 1])$

14 : $returnstate$

15 : $endfunction$

Figure 3 – Pseudo-code pour le chiffrement

3.1.4 L'AES sous forme de système d'équations

En effet, le système d'équations quadratiques multivariées que nous allons présenter est implémenté sur ordinateur et il est donc possible de le mettre pratiquement en œuvre. Ce système d'équations est décrit par Cid, Murphy et Robshaw .Ils y définissent deux variantes réduites de l'AES notées SR et SR*. Ces deux variantes ont les mêmes paramètres à savoir : n le nombre de tours pour le chiffrement, r et c définissent respectivement le nombre de lignes et de colonnes de la table d'entrée et e la taille d'un mot en bits. SR et SR* diffèrent par la forme du dernier tour. Ces deux environnements travaillent sur des blocs de taille $r \times c \times e$ et le tableau d'états est un tableau $r \times c$ contenant des mots de e *bits*. Enfin, la version complète de l'AES-128 est décrite par : $SR^*(10, 4, 4, 8)$.

Implémentation de SR

Afin d'étudier les systèmes d'équations précédemment décrits, nous allons utiliser le logiciel Sage. Sage est l'acronyme de Software for Algebra and Geometry Experimentation, c'est un logiciel open source, basé sur le langage python.[17]

Sage est fourni par défaut avec un certain nombre de modules comprenant notamment des outils pour travailler sur des constructions algébriques. Ainsi, le code suivant montre comment créer un anneau polynomial univarié sur l'anneau Z

```
sage: Z.<x>=ZZ[]
sage: Z
sage: Univariate Polynomial Ring in x over Integer Ring
```

Une fois cet anneau créé, plusieurs opérations sont possibles comme la génération d'un membre aléatoire de l'anneau, l'affichage de la caractéristique de l'anneau ou encore de sa finitude :

```
sage: Z.randomelement()
2-x^2-x+3
sage: Z.characteristic()
40
sage: Z.isfinite()
False
```

De la même façon, il est possible de travailler avec des anneaux polynomiaux sur GF(2) :

```
sage: Z.randomelement()
2-x^2-x+3
sage: Z.characteristic()
40
sage: Z.isfinite()
False
sage: A.<x>=GF(2)[]
sage: A
UnivariatePolynomialRinginxoverFiniteFieldofsize2
sage: (x^2+1).isirreducible()
False
sage: (x^3+x+1).isirreducible()
True
sage: x.degree()
1
sage: x.list()
[0,1]
sage:
```

En 2007, Martin Albrecht [2] a commencé à développer un module pour le logiciel Sage implémentant l'environnement SR détaillé ci-dessus. Commençons par travailler avec SR(1, 1, 1, 4), c'est-à-dire avec un chiffrement AES sur 1 tour avec un tableau d'états de 1 colonne et de 1 ligne et un mot de 4 bits. Le code suivant permet de construire SR(1, 1, 1, 4) et d'afficher le résultat :

```
sage: sr=mq.SR(1,1,1,4)
sage: sr
SR(1,1,1,4)
sage:
```

À partir de là, nous pouvons afficher les variables :

```
sage: printsr.R.reprlong()
PolynomialRing
BaseRing: FiniteFieldinaofsize2^4
```

```
Size:20 Variables
Block0: Ordering: degrevlex
Names: k100, k101, k102, k103, x100, ...
sage:
```

Puis, calculer et afficher le polynôme irréductible :

```
 sage: sr.basering().polynomial()
a^4+a+1
sage:
```

Ou encore, afficher la S-Box :

```
sage: sr.sbox()
(6,11,5,4,2,14,7,10,9,13,15,12,3,1,0,8)
sage:
```

Et enfin, calculer le système d'équations polynomiales multivariées :

```
sage: sr.polynomialsystem()
(PolynomialSystemwith40Polynomialsin20Variables,
{k003: a, k002: a^2+1, k001: a+1, k000: a^2})
sage:
```

Comme nous l'avons vu plus haut, dans notre description de l'environnement SR, l'AES-128 est donné par : SR*(10, 4, 4, 8). En utilisant le module SR de Sage, nous obtenons les données suivantes : Construction de SR*(10, 4, 4, 8) et affichage du résultat :

```
sage: sr=mq.SR(10,4,4,8,star=True)
sage: sr
SR*(10,4,4,8)
sage: sr.basering()
FiniteFieldinaofsize2^8
sage:
```

Calcul et affichage du système polynomial correspondant :

```
sage: sr.polynomialsystem()
k001507: a^7+a^6+a,
 k001506: a^6+a^5+a+1,
 k001505: a^5+a^4+a^2+a,
 k001504: a^5+a^4+a^3+1,
 k001503: a^5+a^3+a^2+a+1,
k001502: a^7+a^6+a^4
 k001501: a^6+a^5+a^2+a,*
k001500: a^5+a^4+a^2+1,
 ....
sage:
```

3.2 implantation de l'attaque par collision sur AES-128

3.2.1 Analyses

Notation 63. *Base de Grôbner Zéro dimensionnelle pour AES – 128*

Dans le chapitre précédent, il a été montré que pour certains chiffrements, une base de Grôbner zéro dimensionnelle DRL peut être calculée à la main. Cela réduit le problème de récupération de clé pour ces chiffrements par blocs en une conversion de la base de Grôbner.

Dans ce chapitre, nous appliquons la méthode similaire à AES-128. Utiliser d'abord un représentation polynomiale de la S-box AES sur $F = GF(2^8)$ nous montrons comment une base de Grôbner de dimension zéro DRL peut être dérivée dans ce cas. Ensuite nous étudier la signification cryptanalytique de cette base de Grôbner.

Notation 64. *Construction de la base de Grôbner DRL*

Rappelons-nous d'abord l'idée de base de la méthode utilisée dans le précédent chapitre pour dériver une base de Grôbner DRL sans réduction polynomiale. Laisser la S-box d'un chiffre soit donnée par un polynôme en entrée, puis le terme de tête de ce polynôme w.r.t. tout ordre de terme de degré total est une puissance de la variable d'entrée. Puisque toutes les entrées des S-box sont différentes, la tête les termes des polynômes correspondants sont premiers par paires. Cependant par le transformation linéaire du chiffre ces polynômes sont mélangés. En cas d'inversion la transformation linéaire nous obtenons des polynômes avec tête première par paire termes, alors par le théorème l'ensemble de ces polynômes est une base de Grôbner.

Considérons maintenant AES-128. Pour notre méthode nous ne pouvons pas utiliser les représentations algébriques sous la forme d'un système d'équations quadratiques, car dans ce cas les termes de tête des polynômes non linéaires ne sont pas univariés, et donc pas de prime par paire. Ainsi nous construirons une base DRL Grôbner pour AES en utilisant la représentation polynomiale donnée précedement.

La première étape de la construction est la suivante. Afin d'avoir polynonome avec des termes principaux par paires dans tous les cycles de chiffrement, nous réécrirons le système d'équations pour le ième tour ($1 \leq i \leq 9$) en utilisant le matrice inverse D^{-1} :

$$D^{-1} \cdot \begin{pmatrix} x_{i,0}+k_{i,0} & x_{i,4}+k_{i,4} & x_{i,8}+k_{i,8} & x_{i,12}+k_{i,12} \\ x_{i,1}+k_{i,1} & x_{i,5}+k_{i,5} & x_{i,9}+k_{i,9} & x_{i,13}+k_{i,13} \\ x_{i,2}+k_{i,2} & x_{i,6}+k_{i,6} & x_{i,10}+k_{i,10} & x_{i,14}+k_{i,14} \\ x_{i,3}+k_{i,3} & x_{i,7}+k_{i,7} & x_{i,11}+k_{i,11} & x_{i,15}+k_{i,15} \end{pmatrix}$$

$$+ \begin{pmatrix} S(x_{i-1,0}) & S(x_{i-1,4}) & S(x_{i-1,8}) & S(x_{i-1,12}) \\ S(x_{i-1,5}) & S(x_{i-1,9}) & S(x_{i-1,13}) & S(x_{i-1,1}) \\ S(x_{i-1,10}) & S(x_{i-1,14}) & S(x_{i-1,2}) & S(x_{i-1,6}) \\ S(x_{i-1,15}) & S(x_{i-1,3}) & S(x_{i-1,7}) & S(x_{i-1,11}) \end{pmatrix} = 0$$

Au dernier tour, la transformation *MixColumns* est omise et chaque équation a les termes d'un polynôme S-box :

$$\begin{aligned} &S(x_{9,0}) + x_{10,0} + k_{10,0} = 0 \\ &S(x_{9,0}) + x_{10,0} + k_{10,0} = 0 \\ &S(x_{9,1}) + x_{10,9} + k_{10,9} = 0 \\ &\qquad\qquad \cdots \\ &S(x_{9,15}) + x_{10,4} + k_{10,4} = 0 \end{aligned}$$

Il est facile de voir que pour les polynômes de ces systèmes l'ensemble des termes principaux *w.r.t.* tout ordre total des termes de degré est $\{x_{i,j}^{254} : 0 \leq i \leq 9, 0 \leq j \leq 15\}$,et aucun polynôme n'a le même terme de tête.

De plus, dans le polynôme d'une équation de texte chiffré $x_{10,j} + c_j = 0$ avec $0 \leq j \leq 15$ le terme de tête est $x_{10,j}$, et il n'a pas de commun non trivial diviseur avec tout autre terme de tête. Les termes $x_{0,j}$et $k_{0,j}$ d'un texte en clair polynôme $x_{0,j} + k_{0,j} + p_j$ ont le même degré. On choisit un ordre des termes tel que $x_{0,j} < k_{0,j}$ *pour tout* $j = 0, 15$.

Considérons enfin les équations clés du programme :

$$\begin{pmatrix} k_{i,1} \\ k_{i,2} \\ k_{i,3} \\ k_{i,3} \\ k_{i,4} \\ \vdots \\ k_{i,15} \end{pmatrix} = \begin{pmatrix} k_{i-1,0} + S(k_{i-1,13}) + \zeta^{i-1} \\ k_{i-1,1} + S(k_{i-1,14}) \\ k_{i-1,2} + S(k_{i-1,15}) \\ k_{i-1,3} + S(k_{i-1,12}) \\ k_{i-1,4} + k_{i,0} \\ \vdots \\ k_{i-1,15} + k_{i,11} \end{pmatrix}$$

où $1 \leq i \leq 10$. Nous voyons que la condition que tous les termes de tête sont par paires prime ne tient plus. Par exemple, le terme de tête du polynôme $S(k_{0,13}) + k_{1,0} + k_{0,0} + 01$ est $k_{0,13}^{254}$, et il est divisible par le terme de tête de $x_{0,13} + k_{0,13} + p_{13}$. En utilisant le polynôme S_0 pour la S-box inverse, nous réécrivons les équations clés du programme comme :

$$\begin{pmatrix} k_{i,1} \\ k_{i,2} \\ k_{i,3} \\ k_{i,3} \\ k_{i,4} \\ \vdots \\ k_{i,15} \end{pmatrix} + \begin{pmatrix} k_{i-1,0} + S(k_{i-1,13}) + \zeta^{i-1} \\ k_{i-1,1} + S(k_{i-1,14}) \\ k_{i-1,2} + S(k_{i-1,15}) \\ k_{i-1,3} + S(k_{i-1,12}) \\ k_{i-1,4} + k_{i,0} \\ \vdots \\ k_{i-1,15} + k_{i,11} \end{pmatrix} = 0$$

Si l'ordre du terme fixe est telle que $k_{i,15} > k_{i,14} > \cdots > k_{i,0} > k_{i-1,15} > \cdots > k_{i-1,1} > k_{i-1,0}$

avec $1 \leq i \leq 10$, alors l'ensemble des termes principaux pour les polynômes de nomenclature clé est $\{k_{i,j}^{254}, k_{i,h} : 1 \leq i \leq 10, 0 \leq j \leq 3, and\ 4 \leq h \leq 15\}$.

Ainsi, en utilisant un terme approprié, ordonnez les polynômes du système pour l'AES ont des termes principaux par paires. Par exemple, le DRL l'ordre des termes avec l'ordre suivant des variables satisfait à cette condition :

$$x_{0,0} < ... < x_{0,15} \quad < \quad k_{0,0} .... < k_{0,15}$$

Variables texte en clair Variables clefs initiales

$$k_{1,0} .... < k_{1,15} \quad < \quad ... \quad < k_{10,0} .... < k_{10,15} <$$

Variables clefs premier tour Variables clefs dernier tour

$$x_{1,0} < ... < x_{1,15} \quad < \quad ... \quad < x_{9,0} < ... < x_{9,15} <$$

variables d'état internes de premier tour variables d'état internes de 9° tour

$$x_{10,0} < ... < x_{10,15}$$

variables de texte chiffré

Notons l'ensemble des polynômes obtenus par A, et cet ordre des termes DRL par $<_A$. Alors par le théorème 65 , A est une base de Grôbner relative à $<_A$. De plus, on voit que A remplit la condition du théorème 66, et donc l'idéal $\langle A \rangle$ est de dimension zéro.

Théorème 65. *Soit $G \subset R$ un ensemble fini de polynômes. Si le terme de tête de certains $f, g \in G$ sont premiers, c'est-à-dire $pgcd(HT(f), HT(g)) = 1$, alors $NF(spol(f, g), G) = 0$. En particulier, si tous les éléments de l'ensemble $HT(G)$ sont coprime par paires, alors G est une base de Grôbner.*

Théorème 66. *Soit G une base de Gr¨obner d'un idéal I. Alors $dimI = 0$ ssi pour tout $i = 1, n$ il existe un polynôme $g \in G$ tel que $HT(g) = x_i^{di}$.*

Théorème 67. *Soit $G_1 \subset R$ une base de Gr¨obner w.r.t. un terme d'ordre 1 d'un idéal polynomial de dimension nulle I, et $D = dim(R/I)$. Ensuite, par l'algorithme FGLM, nous pouvons convertir G_1 en une base de Gr¨obner G_2 w.r.t. un ordre des termes $\prec_2$ dans les opérations du champs $O(nD^3)$. [?]*

Exploitation de la base de Grôbner

Dans cette section, nous étudions la signification cryptanalytique de la base de Grôbner A construit ci-dessus. Nous estimons d'abord la complexité d'une conversion de A à une base Lex Grôbner en utilisant l'algorithme FGLM, on trouve alors un invariant sous l'élimination des variables et expliquer pourquoi A ne peut pas être utilisé pour deviner parties de la clé d'un tour.

Complexité des conversions de base de Grôbner

Selon le théorème 67, la complexité d'une conversion de A en base de Grôbner LEX utilisant l'algorithme $FGLM$ dépend du nombre de variables et la dimension de F-espace vectoriel $R/\langle A \rangle$.

L'ensemble A se compose de 200 polynômes de degré 254 et 152 polynômes linéaires en 352 variables, $x_{i,j}, k_{i,j}$ *avec* $0 \leq i \leq 10$ *et* $0 \leq j \leq 15$. Soit $R = F[V]$, où $V = \{x_{i,j}, k_{i,j} : 0 \leq i \leq 10, 0 \leq j \leq 15\}$. Par le lemme précedent la dimension d'espace vectoriel de $R/\langle A\rangle$ est :

$$dim(R/\langle A\rangle) = 254^{200} \approx 2^{1598}$$

Ce nombre est très énorme. Bien que dans le théorème 67, une limite supérieure du temps d'exécution est donné, il n'y a aucune raison de s'attendre à ce que l'algorithme FGLM soit efficace dans ce cas.

Le nombre de variables peut être réduit par élimination. Cependant, le la dimension d'espace vectoriel de l'idéal est invariante sous l'élimination de tout variables à l'exception des variables clés du dernier tour. Pour le prouver, nous avons besoin du proposition suivante :

Proposition 68. *1 Soit* $I^{'}$ *un idéal de dimension nulle de* $R^{'} = K[x_1, ..., x_n]$, I *un idéal de* $R = R[x_{n+1}]$ *et* $I^{'} = I \cap R'$. *Puis* $dimR/I = dimR^{'}/I^{'}$ *ssi il existe un polynôme* $g \in R^{'}$ *tel que* $x_{n+1} + g \in I$.

Démonstration. La dimension d'espace vectoriel d'un idéal ne dépend pas d'un terme commande. Fixons un ordre lexicographique des termes tel que xn + 1 soit le □

la plus grande variable. Soit $RT(I)$ et $RT(I^{'})$ définis comme suit :

$$RT(I) = \{t \in T(R) : s - t \; for \; all \; s \in HT(I)\}$$

$$RT(I^{'}) = \{t \in T(R^{'}) : s - tforalls \in HT(I0)\} \subset RT(I)$$

Par la proposition 1 , $dimR/I = \#RT(I)$. Il suffit donc de prouver que $\#RT(I) = \#RT(I^{'})$. Puisque $x_{n+1} - t$pour tout $t \in T(R^{'})$, l'égalité $RT(I) = RT(I^{'})$ vaut ssi $x_{n+1} \in HT(I)$, c'est-à-dire qu'il existe un polynôme $g \in R^{'}$ tel que $x_{n+1} + g \in I$.

Corollaire 69. *Soit* $R^{'} = F[k_{0,10}, ..., k_{15,10}]$ *et* $\left\langle A^{'} \right\rangle = \langle A\rangle \cap R^{'}$. *ensuite*$dimR^{'}/\left\langle A^{'} \right\rangle = dimR/\langle A\rangle = 254^{200}$.

Démonstration. Par récurrence en utilisant la proposition 68. Donc, même éliminer toutes les variables mais les variables clés de chiffrement ne réduit pas la complexité de la conversion de la base de Grôbner en un ordre de terme approprié pour la récupération de clé. □

Problème d'adhésion idéal et clés de test

La réduction modulo sur une base de Grôbner est un moyen simple de vérifier l'appartenance idéale d'un polynôme. Puisque $\langle A\rangle$ est un idéal de dimension zéro, il contient des polynômes univariés pour toutes les variables d'octets clés. Ces polynômes peuvent être facilement distingués des autres en utilisant la base de Grôbner construite. De plus, chacun d'eux a évidemment un zéro à $\mu^i \in F$, où μ^i est la valeur correcte de l'octet clé correspondant. Cependant, le polynôme est plus difficile que

$$k^i + \mu^i,$$

et il ne peut pas être facilement deviné. En effet, le système polynomial construit a des solutions sur la fermeture du champ au sol, ce qui signifie que nous avons pour tester un polynôme $p = q \cdot \prod_j (k_i + C_j)^{t_j}$, où tous $C_j \in F$ sont clés octets candidats et q est le produit de polynômes irréductibles sur F. [**?**] De plus, le degré de p peut être aussi grand que $dim(R/\langle A \rangle)$. On voit ça la dimension de $R/\langle A \rangle$ joue ici encore un rôle important : elle égale la nombre de solutions au cours de la fermeture du champ au sol. Comme il a été montré dans la section précédente, ce nombre est obscénément grand. Pour éliminer tous les points de la variété qui ne se trouvent pas dans F, on peut adjoindre l'ensemble

$$F = \{v^{256} + v : pour\ tout\ v \in R\}$$

de tous les polynômes des équations de champ à A. Malheureusement, dans ce au cas où nous n'avons plus de base Grôbner. Ce que nous devons faire ici consiste à calculer l'intersection de deux variétés ; ceci est généralement réalisé par calculer la base de Grôbner de la somme des idéaux correspondants. nous avoir une base de Grôbner A

décrivant AES et un deuxième ensemble de polynômes F, qui forme évidemment une base de Grôbner par rapport au même ordre des termes $\prec_A$aussi. On ne sait cependant pas comment exploiter la propriété de base Grôbner du contribution.

Dans ce chapitre, nous utilisons les bases de Grôbner pour améliorer les attaques de collision par canal auxilaire sur le Cryptosystème AES.

Les attaques de collision auxiliaire ont été introduites et appliquées à AES. Ces attaques fonctionnent en deux étapes. Un attaquant applique d'abord analyse de puissance différentielle à une implémentation physique d'un cryptosystème pour extraire des informations secrètes supplémentaires sur ce système. À la seconde étape l'attaquant récupère la clé secrète en utilisant les informations dérivées. dans le cas d'AES, l'attaquant détecte en comparant les courbes de consommation électrique pour les opérations S-box, si deux octets d'entrée de ces S-box sont égaux.

Dans l'attaque de base proposée dans , seules les collisions se produisant dans l'entrée octets du deuxième tour de différentes exécutions AES à des positions d'octet égales sont utilisés. Il a été montré que l'égalité des entrées aux différentes Sbox peut être détectée. Ces collisions appelées collisions internes généralisées peut être décrit comme un système d'équations polynomiales sur $GF(2^8)$ en clé variables d'octet. Seuls les systèmes qui peuvent être résolus par algèbre linéaire méthodes ont été envisagées. Pour améliorer ces résultats, dans nos attaques non linéaires les collisions ainsi que les non-collisions sont prises en compte. Ici nous ne discuter des techniques de canal secondaire et se concentrer sur le problème de récupération de clé sous l'hypothèse selon laquelle les collisions internes généralisées, telles que décrites dans, peuvent être détecté. Pour plus de détails sur l'analyse de puissance différentielle, y compris dans le cas AES.

Par une ou plusieurs exécutions AES, une collision interne généralisée se produit chaque fois que les octets d'entrée de deux S-box sont égaux. Étant donné que chaque tour d'un cryptage AES a 16 S-box, il existe une grande variété de collisions possibles. toutefois seules certaines de ces collisions peuvent être efficacement exploitées. Dans ce

qui suit, nous décrire plusieurs types de collisions utiles et comment les utiliser pour récupérer la clé secrète AES complète. Les deux premières sous-sections rappellent les attaques de collision sur AES de . Puis le linéaire et le non-linéaire les collisions utilisées dans nos attaques de collisions algébriques ainsi que les non-collisions sont décrit.

Supposons que $m \geq 2$ textes en clair notés $P^{(e)} = (p_0^{(e)}, ..., P_{15}^{(e)})$ *avec* $1 \leq e \leq m$ sont cryptés en utilisant AES-128 avec une clé secrète fixe, $K = (k_0, ..., K_{15})$. Notons $b_{i,j}^{(e)}$ le jème octet de l'état interne avant la ième application de la transformation *SubBytes* pour eth AES run, et par $\mathrm{k}_{i,j}$ le jème octet de la ième clé ronde, où $0 \leq i \leq 9et0 \leq j \leq 15$. en particulier, on a $\mathrm{k}_{0,j} = \mathrm{k}_j$ et $\mathrm{b}_{0,j}^{(e)} = p_j^{(e)} + k_j$ pour tout j. Nous supposons aussi que tous les textes en clair sont connus d'un attaquant.

Notation 70. *Collisions internes*

Schramm, Leander, Felke et Paar ont proposé des attaques par collision à canal auxilaiare sur l'AES basées sur la détection de collisions internes. Une collision interne, se produit, si $\mathrm{b}_{i,j}^{(d)} = \mathrm{b}_{i,j}^{(e)}$ *pour certains* i, j *et* $d \neq e$. On voit que les collisions entre octets du premier tour ne donnent aucune information sur la clé secrète. En effet, $b_{0,j}^{(d)} = p_j^{(d)} + k_j$ et $b_{0,j}^{(e)} = p_j^{(e)} + k_j$ sont égal ssi $p_j^{(d)} = p_j^{(e)}$. Chaque octet de n'importe quel état après le deuxième tour dépend sur tous les octets de la clé secrète, tandis que tout $b_{1,j}^{(e)}$ dépend du quatre octets de la clé du premier tour et un octet de la clé du deuxième tour. Pour cette raison uniquement les collisions interne entre octets du second tour sont utilisées dans pour attaquer AES.

Supposons que $\mathrm{b}_{1,0} = \mathrm{b}_{1,0}^{'}$ pour deux exécutions AES. Puisque

$$b_{1,0}^{(e)} = k_{1,0} + 02{\cdot}S(p_0^{(e)}k_0) + 03{\cdot}S(p_5^{(e)} + p_5) + S(p_{10}^{(e)}) + k_{10}) + S(p_{15}^{(e)} + k_{15})$$

pour tout $e = 1, m$, on a $b_{1,0}$ et $b_{1,0}^{'}$ se heurtent ssi

$$02{\cdot}S(p_0 + k_0) + 03{\cdot}S(p_5 + k_5) + S(p_{10} + k_{10}) + S(p_{15} + k_{15}) = 02{\cdot}S(p_0^{'} + k_0) + 03{\cdot}S(p_5^{'} + k_5) + S(p_{10}^{'} + k_{10}) + S(p_{15}^{'} + k_{15}). \qquad (1)$$

Si $(p_0, p_5, p_{10}, p_{15}) \neq (p_0^{'}, p_5^{'}, p_{10}^{'}, p_{15}^{'})$, alors (1) décrit une relation non triviale entre quatres octets de la clé secrète et peut être utilisé pour réduire le jeu de clés possibles. Equations similaires dans $\{k_0, k_5, k_{10}, k_{15}\}, \{k_3, k_4, k_9, k_{14}\}, \{k_2, k_7, k_8, k_{13}\} ou \{k_1, k_6, k_{11}, k_{12}\}$ sont dérivés de collisions internes entre les octets du second tour à d'autres positions d'octet.

Par définition, mettez $C(\alpha, \beta, k_0, k_1, k_2, k_3) = 02{\cdot}(S(\alpha + k_0) + S(\beta + k_0)) + 03{\cdot}(S(\alpha + k_1) + S(\beta + k_1)) + S(\alpha + k_2) + S(\beta + k_2) + S(\alpha + k_3) + S(\beta + k_3)$

pour tout $\alpha, \beta, k_0, k_1, k_2, k_3 \in GF(2^8)$. Il est évident que $C(\alpha, \beta, k_0, k_1, k_2, k_3) = 0$ ssi $C(\alpha+\beta, 0, k_0+\beta, k_1+\beta, k_2+\beta, k_3+\beta) = 0$. L'attaque optimisée donné dans fonctionne comme suit. Pour tout $\delta \in GF(2^8)\backslash\{0\}$ l'ensemble $T_\delta = \{(k_0, k_1, k_2, k_3) \in GF(2^8)^4 : C(\delta, 0, k_0, k_1, k_2, k_3) = 0\}$ est précalculé et stocké. Chaque ensemble contient en moyenne 224 éléments. Le nombre d'éléments stockés peut être réduit approximativement d'un facteur de 32 en utilisant la propriété suivante de T_δ.

Lemme 71. *Si* $(k_0, k_1, k_2, k_3) \in T_\delta$ *pour certains* $\delta \in GF(2^8)\backslash\{0\}$, *alors* $(k_0+\delta_0, k_1+\delta_1, k_2+\delta_2, k_3+\delta_3) \in T_\delta$,

$$(k_0+\delta_0, k_1+\delta_1, k_3+\delta_3, k_2+\delta_2) \in T_\delta, \quad \text{où } \delta_0, \delta_1, \delta_2, \delta_3 \in \{0, \delta\}.$$

De plus, pour dériver la clé secrète, un attaquant faire entrer différents textes en clair sous la forme de $(\alpha_e, ..., \alpha_e)$ de valeurs aléatoires $\alpha_e \in GF(2^8)$ à un AES module. Pour chaque texte en clair, l'attaquant mesure et stocke la puissance courbes de consommation pour les périodes de temps, où $b^{(e)}_{1,0}, ..., b^{(e)}_{1,15}$ sont traités.

Ensuite, on recherche les collisions internes dans chaque octet en comparant les courbes de puissance correspondantes. Pour détecter les collisions, différentes méthodes peuvent être utilisé, comme les différences carrées, la corrélation croisée, l'analyse d'ondelettes. Si pour une paire (α_e, α_d) une collision interne sont détectées, alors la bonne valeur de quatre octets de la clé secrète appartiennent à l'ensemble

$$\{(k_0+\alpha_e, k_1+\alpha_e, k_2+\alpha_e, k_3+\alpha_e) : (k_0, k_1, k_2, k_3) \in T_{\alpha_e+\alpha_d}\}.$$

Les octets clés correspondant à la collision interne à la ième position d'octet de le deuxième tour donné dans le tableau 1. On voit que toutes les collisions dans les octets d'une colonne fournit un ensemble de valeurs possibles des mêmes quatre octets clés.

D'après , l'intersection de ces ensembles n'a qu'un seul élément après environ quatre de ces collisions. Donc environ 16 collisions (quatre collisions pour chaque colonne) sont nécessaires pour récupérer la clé secrète complète. S'il y en a plus d'un candidat clé, l'attaquant répète la procédure pour dériver des collisions supplémentaires ou teste ces candidats en utilisant des paires connues de texte en clair et de texte chiffré.

Soit Pr_m la probabilité que pour m textes clairs aléatoires, au moins une collision interne se produit dans un seul octet fixe. Evidemment, $Pr_m = 1$ si $m > 256$,

et pour $2 \leq m \leq 256$ on a

$$Pr_m = 1 - \prod_{i=0}^{m-1}(1 - i/2^8)$$

Puisque $Pr_m > (0,5)^{1/16}$ pour tout $m \geq 40$, après 40 mesures l'attaquant a le nombre requis de collisions internes au moins dans la moitié des cas.

Collisions internes généralisées linéaires

Le concept de collisions internes généralisées a été proposé par Bogdanov dans . Une collision interne généralisée se produit, si $b^{(d)}_{i,j} = b^{(e)}_{r,s}$ d'environ deux différentes applications S-box, c'est-à-dire $(i, j, d) \neq (r, s, e)$. Les collisions entre Les octets du premier tour $(i = r = 0)$ sont appelés linéaires. Les collisions linéaires

avec $j = s$ est trivial car ils se produisent ssi $p^{(d)}_j = p^{(e)}_j$, et donc ils peuvent être rejeté. Si $b_{0,j} = b'_{0,s}$ avec un $j \neq s$, on a $k_j + k_s = p_j + p'_s$, et k_j est connu ssi k_s est

connu. Ainsi un ensemble de collisions linéaires généralisées peut être décrit comme un système d'équations linéaires sur $GF(2^8)$en clé secrète variables d'octet :

$$\mathrm{S}: \left\{ \begin{array}{c} k_{j_1} + k_{j_2} = \Delta_1 \\ k_{j_3} + k_{j_4} = \Delta_2 \\ ..... \\ k_{j_{2n-1}} + k_{j_{2n}} = \Delta_n \end{array} \right\}$$

Ici, tout Δ_i est la somme de deux octets connus en clair. Notez que dans le système ci-dessus il y a des équations pas nécessairement pour les 16 octets clés. De plus, il a été montré que ce système n'a jamais une solution unique. Soit K_S un ensemble de toutes les variables libres et manquantes pour S. On a donc $d_S = \#K_S \geq 1$ pour tout système. Comme dans ce cas il y a 2^{8d_S} clés candidats, la bonne clé est identifiée à l'aide d'une paire connue de texte en clair et de texte chiffré. La dépendance de d_S sur le nombre de mesures a été analysé. .

Ainsi, en utilisant des attaques de collision linéaires, on peut dériver la clé secrète après 5 mesures en $2^{45,5}$ pas en moyenne avec une probabilité de 0,548, tandis qu'avec 6 mesures l'attaque fonctionne en $2^{37,15}$ étapes et a une probabilité de succès de $0,85$. On voit aussi qu'après 11 mesures l'attaque hors ligne attendue la complexité est d'environ $2^{12,11}$ et pratiquement tous les systèmes peuvent être résolus.

Collisions internes généralisées non linéaires

Pour améliorer les résultats des attaques de collision ci-dessus, nous considérons les collisions linéaires en combinaison avec d'autres types de collisions internes généralisées. Si les octets d'entrée $b_{i,j}^{(d)}$ et $b_{r,s}^{(e)}$ de deux S-box entrent en collision, nous avons le simple linéaire équation sur $GF(2^8)$:

$$b_{i,j}^{(d)} + b_{r,s}^{(e)} = 0,$$

ce qui correspond à 8 équations linéaires sur $GF(2)$ en variables binaires. Sur le d'autre part, chacun de ces octets dépend d'octets d'un texte en clair et la clef secrète. Cette relation peut être décrite par un système d'équations polynomiales, par exemple, en utilisant l'une des représentations AES données précedement.

Pour tous les octets sauf les entrées du premier tour, le système correspondant n'est pas linéaire, et donc une collision interne généralisée entre $b_{i,j}^{(d)}$ et $b_{r,s}^{(e)}$ avec $i \neq 1$ ou $r \neq 1$ est dit non linéaire. Il est clair que l'on peut dériver un système d'équations pour n'importe quel sous-ensemble de toutes les collisions internes généralisées détectées. L'idée générale des attaques par collision algébrique est d'extraire des informations sur la clé secrète en résolvant l'un de ces systèmes. Dans notre cas, nous utilisons Algorithme 3 avec l'algorithme Faugère F4 pour la recherche de base de Grôbner. Remarque que pour tous les sous-ensembles de collisions, le système correspondant ne peut être résolu efficace même si le nombre de collisions détectées est suffisamment important. Dans nos attaques, nous utilisons deux types de collisions non linéaires, les collisions FS et FL, définies au dessous de. Des systèmes d'équations correspondant à ces collisions sont spécifiés dans la section suivante, et les résultats de l'analyse sont donnés dans la section précedente.

Nous considérons d'abord les collisions qui se produisent dans l'AES entre les octets du deux premiers tours. Nous les appelons des collisions FS. On peut distinguer les trois

suivantes sous-types de collisions FS : collisions linéaires au premier tour, collisions non linéaires entre les deux premiers tours et collisions non linéaires dans le deuxième tour. Chaque collision non triviale du premier sous-type lie linéairement deux octets de la clé secrète, tandis que les autres collisions décrivent relations non linéaires entre quatre octets clés ou plus. Naturellement, on peut également considérer des collisions se produisant entre des octets de les trois premiers, quatre et ainsi de suite. Cependant, dans ces cas, la structure des systèmes polynomiaux obtenus est plus difficile. Nous proposons une solution plus efficace attaque basée sur les collisions entre les octets du premier et du dernier round. Nous appelons de telles collisions FL-collisions. Une collision FL peut être l'une des suivantes les types :

$$b_{0,i}^{(d)} = b_{0,s}^{(e)}, \qquad b_{0,i}^{(d)} = b_{9,s}^{(e)}, \qquad and \qquad b_{9,i}^{(d)} = b_{9,s}^{(e)}, with 0 \leq i, s \leq 15 and 1 \leq d, e \leq m.$$

En comparant les courbes de consommation électrique correspondantes pour la S-box opérations on peut aussi détecter que $b_{i,j}^{(d)} \neq b_{r,s}^{(e)}$ pour certains $0 \leq i, r \leq 10, 0 \leq j, s \leq 15 et 1 \leq d, e \leq m$. Dans ce cas, on dit que $b_{i,j}^{(d)}$, $b_{r,s}^{(e)}$ est une non-collision. Dans la section suivante, nous montrons comment les non-collisions peuvent être utilisées pour améliorer ces attaques par collision. Notez également que quatre autres S-box sont appliquées à chaque tour du programme clé. Les collisions et non-collisions avec ces S-box peuvent être également utilisé dans nos attaques.

3.2.2 Attaques

Représentation algébrique de non-linéaire Collision

Dans cette section, nous décrivons les systèmes d'équations polynomiales pour les collisions FS et FL ainsi que les systèmes combinés.

Collisions FS

Pour les attaques de collision algébriques basées sur des collisions FS, nous utilisons des systèmes de équations quadratiques sur $GF(2)$, qui sont dérivées de la représentation polynomiale de l'AES donnée dans la section précedente. Dans ce cas, un polynôme Le système se compose de deux parties. L'un d'eux décrit le premier tour de la Chiffrement AES pour tous les textes clairs donnés et le premier tour de la planification des clés, tandis que les équations du deuxième sous-système correspondent à tous les détectés Collisions FS. L'ensemble de variables se compose de :

• variables de 128 bits pour la clé initiale ; nous utilisons $k_{i,j}^{(0)}$ pour désigner le jème bit variable pour le ième octet de la clé initiale ;

• variables de 128 bits pour la première touche ronde ; on utilise $k_{i,j}^{(1)}$pour désigner le jth variable binaire pour le ième octet de la première touche ronde ;

• variables de 128 · m bits pour toutes les entrées de la première couche S-box ; on utilise $x_{i,j}^{(r)}$ pour désigne la jième variable de bit pour le ième octet de l'état interne avant la première transformation SubBytes par cryptage de $P^{(e)}$;

• variables de 128 · m bits pour toutes les sorties de la première couche S-box ; nous utilisons $y_{i,j}^{(e)}$ pour désigner la variable jième bit pour le ième octet de l'état interne après la première transformation SubBytes par cryptage de $P^{(e)}$;

• variables de $128 \cdot$ m bits pour toutes les entrées de la deuxième couche S-box ; nous utilisons $z_{i,j}^{(e)}$ pour désigner la variable jième bit pour le ième octet de l'état interne avant la deuxième transformation SubBytes par cryptage de $P^{(e)}$; où m est le nombre d'exécutions AES différentes, $0 \leq i \leq 15, 0 \leq j \leq 7, et 1 \leq e \leq m$.

Pour les collisions FS et m textes clairs connus, chaque système polynomial nous consider est l'union de l'ensemble d'équations suivant :

1. Equations en S-box du premier tour pour toutes les exécutions AES. Chacune de ces équations est quadratique sur $GF(2)$ et n'a que 16 variables, $x_{i,0}^{(r)}, \ldots, x_{i,7}^{(r)}$,

et $y_{i,0}^{(r)}, \ldots, y_{i,7}^{(r)}$ pour quelque$0 \leq i \leq 15, 1 \leq e \leq m$.

2. Equations linéaires qui décrivent la composition des transformations *ShiftRows*, *MixColumnset AddRoundKey* du premier tour pour tous les textes en clair connus. Pour chaque $z_{i,j}^{(e)}$, il y a exactement une équation, et le polynôme de cette équation est la somme de $k_{i,j}^{(1)}$ et d'une combinaison linéaire de $y_{i,j}^{(e)}$

3. Équations du premier tour du calendrier clé. Rappelez-vous que pour exprimer les quatre premiers octets de la clé ronde, les équations quadratiques S-box sont utilisé, tandis que les équations pour les autres bits de clé sont linéaires.

4. Les équations en clair :

$x_{i,j}^{(r)} + k_{i,j}^{(0)} + p_{i,j}^{(r)} = 0$ pour tout i, j, e ; ici $p_{i,j}^{(r)}$ est le jème bit du ième octet de $P^{(e)}$.

5. Equations pour toutes les collisions FS détectées. Pour une collision linéaire dans le premier tour, c'est-à-dire si $b(0d, i) = b(0e, r)$ pour un certain$0 \leq i, r \leq 15, 1 \leq d, e \leq m$, nous obtenir $x(i, jd) + x(r, je) = 0$avec $0 \leq j \leq 7$. Si $b(0d, i) = b(1e, r)$, alors $x(i, jd) + zr, j(e) = 0$ pour tout $0 \leq j \leq 7$; et $zi, (d0) + zr, (e0) = \cdots = zi, (d7) + zr, (e7) = 0$dans le cas $b(d)1, i = b(1e, r)$. De la même manière, on peut décrire le cas de $b = b^{'}$, où b ou b$^{'}$ est une entrée dans une boîte S des deux premiers tours de programme clé.

6. Les équations de champ $GF(2)$ pour toutes les variables utilisées. Le nombre de variables et d'équations peut être réduit comme suit. Pour toute exécution AES, l'état initial après le premier tour est inconnu, et seulement les équations de collision contiennent une information sur la clé secrète dans le système construit. Si un octet d'entrée dans une S-box du deuxième tour n'entre en collision avec aucun autre octet d'entrée, alors les équations linéaires pour le correspondant $z_{i,0}^{(e)}, ..., z_{i,7}^{(e)}$ peuvent être supprimés du système. Combinant le équations linéaires pour tous les autres $z_{i,j}^{(e)}$ avec des équations de collision, nous pouvons réécrire les système sans les variables du second tour. De plus, tout $x_{i,j}^{(e)}$ peut être éliminé en utilisant les équations en clair. De toute évidence, le nouveau système peut être écrit directement pour tout ensemble de collisions FS détectées. On voit aussi que le le nombre d'équations quadratiques du système obtenu ne dépend pas de cet ensemble.

Collisions FL

Dans le dernier cycle du chiffrement AES, la transformation *MixColumns* est pas appliqué, et nous avons

$S(b_{9,j}^{(e)}) = k_{10,\pi(j)} + c_{\pi(j)}^{(e)}$ *for all* $0 \leq j \leq 15, 1 \leq e \leq m$, *where* $C^{(e)} = (c_0^{(e)}, ..., c_{15}^{(e)})$ *is the cipher text and*

$$\pi = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\ 0 & 5 & 10 & 15 & 4 & 9 & 14 & 3 & 8 & 13 & 2 & 7 & 12 & 1 & 6 & 11 \end{pmatrix}$$

est la permutation correspondant à ShiftRows. Supposons que $C^{(1)}, ..., C^{(m)}$ sont connus. Pour les collisions FL, nous obtenons

$$b_{0,i}^{(d)} = b_{0,s}^{(e)}, SSI \qquad k_{0,i} + k_{0,s} = p_i^{(d)} + p_s^{(e)};$$

$$b_{9,i}^{(d)} = b_{9,s}^{(e)}, SSI \qquad k_{10,\pi(i)} + k_{10,\pi(s)} = c_{\pi(i)}^{(d)} + c_{\pi(s)}^{(e)};$$

$$b_{0,i}^{(d)} = b_{9,s}^{(e)}, SSI \qquad k_{0,i} + p_i^{(d)} = S^{-1}(k_{10,\pi(s)} + c_{\pi(s)}^{(e)})$$

On voit que les collisions FL correspondent à des relations entre octets de la clé initiale et la dernière touche ronde. Ces relations peuvent évidemment être exprimées sous la forme d'un système d'équations quadratiques sur $GF(2)$. Maintenant, nous montrons comment dériver un système d'équations quadratiques sur $GF(2^8)$ pour ces collisions. Une est d'utiliser l'expression BES comme décrit dans la section précedente. Toutefois, nous avons 8 variables par octet clé dans ce cas. Nous décrivons un système plus simple, qui n'a que 32 variables.

Il est clair que les collisions FL des deux premiers types peuvent être exprimées comme équations linéaires sur $GF(2^8)$. Considérons une collision FL non linéaire du troisième type. Son expression algébrique est donnée par :

$$S(k_{0,i} + p_i^{(d)}) = k_{10,j} + c_j^{(e)}$$

pour certains $0 \leq i, j \leq 15, 1 \leq d, e \leq m$. Rappelez-vous que l'AES S-box est le composition de l'inverse multiplicatif dans le corps fini $GF(2^8)$, le $GF(2)$ - cartographie linéaire, et l'addition XOR de la constante 63. Rappelons que l'inverse de l'application linéaire de $GF(2)$est donné par le polynôme suivant sur $GF(2^8)$:

$$f(x) = 6ex^{2^7} + dbx^{2^6} + 59x^{2^5} + 78x^{2^4} + 5ax^{2^3} + 7fx^{2^2} + fex^2 + 05x.$$

Par conséquent, nous avons $(k_{0,i} + p_i^{(d)})^{-1} = f(k_{10,j} + c_j^{(e)} + 63) = f(k_{10,j}) + f(c_j^{(e)} + 63)$

Si on remplace $f(k_{10,j})$ par une nouvelle variable $(\overline{k}_{10,j})$ on obtient l'équation quadratique

$(k_{0,i} + p_i^{(d)})(\overline{k}_{10,j}) + f(c_j^{(e)} + 63) = 1$

ce qui est vrai avec une probabilité de $\frac{255}{256}$. La proposition suivante suit :

Solutions à l'équation $S(k_{0,i}+p_i^{(d)}) = k_{10,j}+c_j^{(e)}$coïncide avec les solutions de l'équation $(k_{0,i} + p_i^{(d)})(\overline{k}_{10,j}) + f(c_j^{(e)} + 63) = 1$

sous le changement des variables $\overline{k}_{10,j} = f(k_{10,j})$ avec une probabilité de $\frac{255}{256}$.

De plus, si $k_{10,i} + k_{10,j} = \Delta_{i,d),(j,e)} = c_i^{(d)} + c_j^{(e)}$ pour quelque $0 \leq i, j \leq 15$ et $1 \leq d, e \leq m$, alors on a

$$f(k_{10,i}) + f(k_{10,j}) = \overline{k}_{10,i} + \overline{k}_{10,j} = f(\Delta_{i,d),(j,e)})$$

Ainsi nous dérivons pour les collisions FL le système S d'équations quadratiques sur $GF(2^8)$ en 32 variables $K = \{k_{0,i}, \overline{k}_{10,i}\}$ $0 \leq i \leq 15$. De plus, chaque équation du système résultant S n'a que deux variables. Nous appelons de telles équations binôme. Notez que la dernière touche ronde est connectée à la clé initiale par le calendrier clé AES, cependant les équations pour le calendrier clé ainsi que les équations de champ ne sont pas incluses dans nos systèmes polynomiaux dans le cas des collisions FL. Aussi, nous ignorons les S-box du dernier tour de clé programme.

Systèmes d'équations combinés

Les systèmes d'équations donnés ci-dessus décrivent séparément les collisions FS et FL. Dans ce qui suit, nous montrons que pour une attaque par collision algébrique réussie sur AES, une approche combinée peut également être utilisée. Dans ce cas, les collisions FL doit cependant être exprimé par des systèmes polynomiaux sur $GF(2)$. De plus, équations qui décrivent les collisions entre les S-box des deux derniers rounds sont inclus dans ces systèmes. Ainsi, les systèmes combinés bruts détiennent le équations suivantes sur $GF(2)$:

1. Équations en S des premier et dernier tours, c'est-à-dire équations quadratiques décrivant la relation entre l'entrée et la sortie des S-box à les premier et dernier tours ;

2. Equations correspondant à la transformation linéaire du premier tour ;

3. Équations qui décrivent la transformation linéaire inverse de l'arrondi à côté du dernier (tour 9) ;

4. Équations clés du calendrier pour le premier et le dernier tour ; l'un d'eux décrit la relation entre la clé du premier tour et la clé initiale, et les autres lient les sous-clés de deux derniers tours ; l'intermédiaire les équations de calendrier clés ne sont pas incluses dans les systèmes ;

5. Équations de texte en clair et de texte chiffré ;

6. Équations de collision ; dans ce cas, seule une partie des équations exprime le égalité des entrées à certaines S-box. Ce sont des équations dérivées de une des collisions suivantes

$$b_{0,i} = b'_{0,j} \qquad b_{0,i} = b'_{1,j} \qquad b_{1,i} = b'_{1,j}$$

$$b_{0,i} = b'_{9,j} \qquad b_{1,i} = b'_{9,j}$$

avec $0 \leq i, j \leq 15$. Pour chaque collision sous l'une des formes suivantes

$$b_{0,i} = b'_{8,j} \qquad b_{8,i} = b'_{8,j}$$

$$b_{8,i} = b'_{9,j} \qquad b_{9,i} = b'_{9,j}$$

avec $0 \leq i, j \leq 15$, nous avons 8 équations linéaires en variables pour les sorties des S-box correspondantes. Ici$b_{r,i}$ ou $b'_{r,i}$ peut aussi être une entrée pour un S-box dans la liste des clés, où$r = 0, 1, 8, 9$et $12 \leq i \leq 15$. Remarque que les équations pour les collisions sous la forme $b_{1,i} = b'_{8,j}$ ne sont pas incluses dans le système.

En utilisant des équations linéaires, on peut éliminer les variables qui décrivent les entrées des deux premiers tours et sorties des S-box des deux derniers tours.

Alors les systèmes résultants ont 512 variables pour $K^{(1)}, K^{(2)}, K^{(10)}, K^{(11)}$ ainsi que $128 \cdot m$ variables pour les sorties de la première couche S-box et $128 \cdot m$

variables pour les entrées des S-box du dernier tour, où m est le nombre des paires connues de texte clair / chiffré.

Non-collisions

Un ensemble de non-collisions peut également être décrit comme un système d'équations polynomiales sur $GF(2)$ ainsi que sur $GF(2^8)$. Supposons deux octets $b_1 = \{x_7 x_6 ... x_0\}$ et $b_2 = \{y_7 y_6 ... y_0\}$ n'entre pas en collision, c'est-à-dire $b_1 \neq b_2$. Alors les variables binaires satisfont l'équation suivante sur $GF(2)$:

$$\prod_{i=0}^{7}(x_i + y_i + 1) = 0.$$

L'équation correspondante sur $GF(2^8)$ est donnée par $(b_1 + b_2)^{255} + 1 = 0.$

Le degré de la première équation est égal à 8, et le nombre de termes est exactement 38 = 6561, tandis que l'équation sur $GF(2^8)$ a le degré 255 et 257 termes. Les deux équations semblent inutiles pour les attaques basées sur les bases de Grôbner. Cependant, il existe des applications plus constructives de non-collisions réduisant la recherche de plusieurs octets inconnus. Celles-ci sont spécifiques à la structure des systèmes d'équations non linéaires que nous utilisons et sont expliqués dans la sous-section précedente.

Analyse algébrique des collisions

Dans cette section, nous analysons les systèmes d'équations construits ci-dessus. nous montre combien d'exécutions AES sont nécessaires pour récupérer la clé secrète complète. Nombreuses les moyens d'accélérer le processus de recherche des bases de Grôbner pour les systèmes combinés sont présentés et discutés, y compris des chaînes de variables dans les équations binomiales, les cycles non linéaires et l'optimisation de la recherche à l'aide de non-collisions.

Nombre prévu de collisions

Notons d'abord que $b_1 = b_2 = \cdot \cdot \cdot = b_r$ sera considéré comme $r - 1$ collisions pour tous $r \geq 2$. Puis le nombre de collisions entre $b_1, ..., b_n$ est évidemment égal à $n - d$, où d est le nombre d'éléments différents. Soit $\binom{n}{d}$ le Numéro de Stirling du second type, c'est-à-dire le nombre de façons de partitionner un ensemble de n éléments en d sous-ensembles non vides. Ce nombre est donné par

$$\binom{n}{d} = \frac{1}{n!} \cdot \sum_{i=0}^{d} (-1)^{d-i} \binom{d}{i} i^n$$

Si $d < 2^8$, tous les sous-ensembles peuvent être étiquetés avec différents éléments de $GF(2^8)$ dans $256 \cdot 255 \cdot \cdot \cdot \cdot (256 - d + 1)$ voies. Donc pour n éléments aléatoires de $GF(2^8)$, le nombre moyen de collisions peut être calculé à partir de

$$N(n) = \frac{1}{256^n} \cdot \sum_{d=1}^{n'} (n-d).\binom{n}{d}.\frac{256!}{(256-d)!}$$

où $n' = min(n, 256)$. Nous utilisons cette formule pour estimer le nombre de collisions FS et FL après le cryptage AES de m aléatoire les textes en clair. Dans le cas de collisions FS, l'opération S-box est appliquée 32m fois dans les deux premiers tours de cryptage et 8 fois dans les deux premiers tours de l'horaire clé. Pour les collisions FL, $n = 32m + 4$, puisque les S-box du dernier tour du calendrier clé ne sont pas pris en compte ici. Par combiné approche, on recherche des collisions entre entrées à $64m + 16$ S-box, mais ignore les collisions sous la forme $b_1 = b'_8$, où b_1 et b'_8 sont des octets d'entrée du deuxième et avant-dernier tours respectivement. Le résultat pour $m = 2, 3, 4, 5$ est donnée dans le tableau 5. Notez que certaines collisions peuvent être triviales, c'est-à-dire des collisions se produisant indépendamment de la clé secrète.

Depuis une collision non triviale, on peut tirer des informations sur un octet de clé, au moins 16 collisions sont nécessaires pour récupérer la clé complete, sinon plusieurs octets doivent être devinés. Dans le cas de collisions FL

Measurements		2	3	4	5
Type of Coll.		E_m			
FS	$b_0 = b'_0$	2.35	4.86	8.18	12.27
	$b_0 = b'_1$	4.43	8.68	13.98	20.10
	$b_1 = b'_1$	2.35	4.86	8.18	12.27
	$\sum$	9.13	18.40	30.34	44.64
FL	$b_0 = b'_0$	2.35	4.86	8.18	12.27
	$b_0 = b'_9$	3.96	8.07	13.25	19.28
	$b_9 = b'_9$	1.86	4.15	7.28	11.18
	$\sum$	8.17	17.08	28.71	42.73
Comb.	$\sum$	22.69	56.74	90.04	128.62

Tableau 5 : Nombre attendu de collisions après m mesures, E_m

ainsi que par l'approche combinée, il faut plus de collisions, car le les équations de calendrier clés intermédiaires ne sont pas incluses dans nos systèmes et il y a plus de variables indépendantes. Cependant, il n'est pas nécessaire de savoir la valeur de toutes les variables. Il suffit de trouver les octets clés de l'un ou l'autre des premiers ou dernier tour.

Équations, chaînes et cycles binomiaux

Nous considérons uniquement les équations binomiales dans les variables clés. Ils décrivent linéaire les collisions du premier et du dernier round ainsi que les collisions

FL non linéaires. nous voir que chaque système de collisions FL introduit dans la sous-section précedente comprend de ces équations seulement.

Soit S un système d'équations non linéaires pour les collisions FL. Dans ce cas, l'ensemble de variables est $K=\left\{k_i, k_i^{'}; 0 \leq i \leq 15\right\}$

où k_i et $k_i^{'}$ sont respectivement la clé initiale et la dernière variable clé ronde. Considérons une partition de K

$$K = K_1 \cup \cdots \cup K_n, K_i \cap K_j = \emptyset, i \neq j$$

tel que

1. pour tout $1 \leq i < j \leq n$ et deux variables quelconques $v_i \in K_i$ et $v_j \in K_j$ il n'y a pas d'équation en v_1, v_2 en S ;

2. aucun Ki n'a une partition qui satisfait la première propriété.

Nous disons qu'un sous-ensemble des variables est connecté $w.r.t.S$, si ce sous-ensemble n'a pas de partition qui satisfait la première propriété. Ainsi, chaque K_i est connecté. Ensuite, le système S peut être partitionné en n sous-système isolé S_i correspondant à Ki. Les paires (K_i, S_i) sont appelées chaînes. Evidemment, $S_i = \emptyset$

ssi $Ki = \{v\}$ pour une variable $v \in K$. Dans ce cas, la bonne valeur du la variable v ne peut être que devinée. Si $u, v \in Ki$ pour certains i, alors il existe $v_0 = u, v_1, ..., v_t = v \in K_i$ tel que S_i contienne une équation binomiale $p_1(v_0, v_1) = p_2(v_1, v_2) = \cdots = p_t(v_t - 1, v_t) = 0$

Le cas S_i est un sous-système linéaire a été considéré dans . Nous avons rang $(S_i) = \#K_i - 1$, et si on fixe une variable $v_{i1} \in K_i$, les autres variables $v_{ij} \in Ki$ sont donnée par $v_{ij} = v_{i1} + \Delta_{ij}$ avec $\Delta_{ij} \in GF(2^8)$. Supposons maintenant que Si a un ou plusieurs équations quadratiques, c'est-à-dire $K_i \cap \{k_0, ..., k_{15}\} \neq \emptyset$ et $K_i \cap \{k_0^{'}, ..., k_{15}^{'}\} \neq \emptyset$.

Soit $v \in K_i$. Puisque Ki est connecté, il existe une relation entre v et tout autre variable $x \in Ki$. Ces relations peuvent être exprimées sous forme linéaire ou quadratique

équations à deux variables. En effet, soit $x, y, z \in Ki$, et

$$x \cdot y + \alpha \cdot x + \beta \cdot y + \gamma = 0; x + z = \delta,$$

où $\alpha, \beta, \gamma, \delta \in GF(2^8)$. Si $v + x = \bar{\delta}$, on obtient

$$v \cdot y + \alpha \cdot v + (\beta + \bar{\delta}) \cdot y + (\gamma + \alpha \cdot \bar{\delta}) = 0; v + z = \delta + \delta.$$

Dans le cas $x \cdot v + \alpha^{-} \cdot x + \beta^{-} \cdot v + \gamma^{-} = 0$ on a

$$(v + \alpha^{-})(x \cdot y + \alpha \cdot x + \beta \cdot y + \gamma) + (y + \alpha)(x \cdot v + \alpha^{-} \cdot x + \beta^{-} \cdot v + \gamma^{-}) =$$

$$(\beta + \bar{\beta}) \cdot v \cdot y + (\alpha \cdot \bar{\beta} + \gamma) \cdot v + (\alpha^{-} \cdot \beta + \gamma^{-}) \cdot y + (\alpha^{-} \cdot \gamma + \alpha \cdot \gamma^{-}) = 0,$$

et

$$v \cdot z + \alpha^{-} \cdot z + (\beta^{-} + \delta) \cdot v + (\gamma^{-} + \alpha_{-} \cdot \delta) = 0.$$

Nous voyons que le degré des polynômes S n'augmente pas et est ≤ 2. Par conséquent, une base de Grôbner pour S peut être trouvée rapidement. Voyons maintenant combien de solutions S_i a dans le cas non linéaire. En tant que exemple, si $K_i = \{v, u\}$, et Si a deux équations non linéaires dans u, v, alors v est racine d'une équation quadratique en une variable. Par conséquent, Si a deux solutions dans ce cas. Si $Si \geq 3$, alors la solution est unique. Généralement, on dit que Ki est fortement connecté avec. Si, s'il existe une équation non linéaire $e \in Si$ de telle sorte que K_i est connecté w.r.t. Si e. De telles chaînes sont appelées cycles. Il on peut montrer que dans ce cas S_i a au plus deux solutions. De plus, le la solution est unique, si K_i est fortement lié à $w.r.t.$ $S_i \backslash \{e\}$.

Ainsi le nombre de solutions de l'ensemble du système S est égal à $Qmi = 12qi$, où $qi \leq 1$,si Ki est fortement connexe, et $qi = 8$ sinon. Notez que à la fois le nombre de candidats clés initiaux et le nombre de clés de la dernière ronde les candidats peuvent être inférieurs au nombre de toutes les solutions. Clairement, c'est assez pour considérer l'un de ces sous-ensembles, le minimum. S'il a aussi plus que ça une solution, la clé correcte peut également être détectée en utilisant le programme de clé en tant que paires connues de texte brut / texte chiffré. Résultats expérimentaux pour les collisions FL sont donnés dans la section précedente

En outre, considérez un système combiné. Il a un sous-système qui se compose des équations de collision FL réécrites sur $GF(2)$. Cependant, dans ce cas, le les équations non linéaires ne sont pas binomiales. Cela a un impact dramatique sur le temps d'exécution du calcul de base de Gr¨obner. Pour atténuer cet impact, nous résoudre tous les cycles sur $GF(2^8)$, puis remplacer la solution obtenue par la variables de bit correspondantes dans le système combiné, et résolvez le reste de équation sur $GF(2)$.

Si un système sur $GF(2)$ ne peut être résolu dans un délai raisonnable, on peut devinez plusieurs octets clés et essayez à nouveau de résoudre le système. Ici les chaînes qui ne sont pas des cycles sont utilisés. Comme indiqué ci-dessus, les chaînes possèdent la propriété qu'un élément de la chaîne définit de manière unique tous les autres éléments du chaîne. La stratégie appliquée ci-dessous est de trouver les chaînes les plus longues et de deviner un octet dans chacun d'eux. Cela permet de déterminer le nombre maximal de les inconnues dans le système par le même coût de deviner.

Accélération grâce aux non-collisions

Dans le cas des systèmes combinés ainsi que des systèmes pour les collisions FS, nous devinons parfois h octets avant de résoudre comme décrit dans la sous-section précédente.

Cela signifie que nous devons résoudre les systèmes résultants $28 \cdot$ h pour récupérer le secret clé. Dans les attaques les plus pratiques, h peut être 1 ou 2. Nous montrons maintenant comment le le nombre de suppositions peut être réduit en utilisant les non-collisions introduites dans la sous-section precedente. Au lieu de construire des équations non linéaires implicites de degré 8, nous utiliser explicitement les non-collisions de la manière suivante.

Soit $C_1, ..., C_h$ désigne les h plus longues chaînes de variables induites par les équations binomiales en question, chacune étant de longueur $|Ci| = l_i$ avec $1 \leq i \leq h$, et $C_i \cap C_j = \emptyset$ pour i 6 = j. Les variables de la chaîne Ci sont notées ci, j pour $1 \leq j \leq li$. Chacun d'eux correspond soit à l'octet clé initial, soit au dernier octet clé. l'ensemble

$B_{i,j} = (p_r^{(1)} + c_{i,j}, p_r^{(2)} + c_{i,j}, ..., p_r^{(m)} + c_{i,j})$,

$if\ c_{ij} = k_{0,r}, and \quad B_{i,j} = (S^{-1}(c_r^{(1)} + c_{i,j}), S^{-1}(c_r^{(2)} + c_{i,j}), ..., S^{-1}(c_r^{(m)} + c_{i,j}))$

dans le cas de $c_{i,j} = k_{10,r}$; ici m est le nombre de mesures avec paires de texte brut / texte chiffré. En d'autres termes, nous connectons chaque chaîne avec le tableau des valeurs des octets d'entrée dans les S-box correspondantes. Pour toute supposition $(c_{1,1}, c_{2,1}, ..., c_{h,1})$, on obtient tous les autres éléments de chaînes en utilisant la collision équations et définissez les tableaux comme ci-dessus. Ensuite, nous pouvons enregistrer toutes les collisions apparaissant dans ces tableaux et comparez la liste résultante avec l'ensemble de tous les collisions ainsi que les non-collisions détectées par mesure. Il est clair que le liste contient les vraies collisions, puisqu'elles sont utilisées pour dériver des éléments du Chaînes. Cependant, si la supposition est fausse, il est possible que dans les tableaux, nous ayons une collision entre deux S-box, où la non-collision a été détectée en réalité.

Cela permet d'optimiser la recherche d'évaluations de chaînes. La procédure peut être formalisé à l'aide de l'algorithme 5 :

5 Sieving guesses with non-collisions and non-linear cycles

Require : h chains C1, . . . , Ch of lengths l1, . . . , lh with 28h possible evaluations, and the list L of all collisions detected for these chains

1 : *pour tout guess* $(c1, 1, \ldots, ch, 1) \in \{0, \ldots, 28h - 1\}$ *do*

2 : pour tout chain i = 1 : h do

3 : Evaluate Bi,1

4 : pour tout chain variable ci,j j = 2 : li do

5 : Evaluate ci,j and Bi,j using chain equations

6 : end for

7 : end for

8 : pour tout (bi,j,k, br,s,t) of l(l-2 1) pairs of table elements in T =

((B1,1, . . . , B1,l1), . . . , (Bh,1, . . . , Bh,lh)) /* where l = m · P li */ do

9 : if bi,j,k = br,s,t then

10 : Verify if the corresponding collision ε lies in L

11 : si $\varepsilon \in$ L then

12 : Go to 1 (contradiction is detected)

13 : end if

14 : end if

15 : end for

16 : Output the guess (c11, . . . , ch1) as a candidate evaluation of the chains

17 : end for.

3.2.3 Résultats expérimentaux

Résolution d'équations pour les collisions FS

L'application simple de l'algorithme Faugère F4 au système construit dans la sous-section précedente donne des résultats supérieurs à ceux de. Celles-ci sont résumées dans le tableau 7

Le système d'équations non linéaires est considéré sur $GF(2)$. Pour m entrées (m mesures), il y a 128 variables de la première sous-clé $K^{(1)}$, 128 variables de la deuxième sous-clé $K^{(2)}$et 128$\cdot$ m variables intermédiaires pour le bits de sortie de la première couche ronde S-box. Les équations indépendantes de la collision comprennent 16$\cdot$ $t\cdot$ m équations quadratiques sur $GF(2)$reliant les entrées et sorties des premièrs tours de S-box, et 4$\cdot$ t quadratique et 12$\cdot$ 8 = 96équations linéaires reliant $K^{(1)}$et $K^{(2)}$en utilisant les relations de nomenclature clés.

Ici t est le nombre d'équations quadratiques utilisées pour exprimer la S-box AES, nous avons t = 23 ou 39. Chacun des trois types de FScollisions ajoute 8 équations linéaires au système, ce qui donne 8 $\cdot$ c équations si c collision s'est produite. De plus, les équations de champ pour toutes les variables sont inclus dans le système.

Le système est résolu de la manière suivante. Le système est d'abord passé à l'algorithme F4 sans modifications. S'il n'est pas résoluble, on devine le plus grand composant linéaire connecté et tente à nouveau de résoudre le système. Comme critère de solvabilité, le coût mémoire a été utilisé. La raison pour laquelle nous utilisons

ce critère est le suivant. Supposons pour certains systèmes la base de Gr¨obner le calcul a un coût de mémoire élevé, cela signifie que dans les étapes internes

de l'algorithme F4, les matrices de Macaulay à transformer sont grand. Ainsi, le calcul de base de Gr¨obner pour ce système doit également avoir un coût de temps élevé. D'autre part, il existe des systèmes qui peuvent être résolus beaucoup plus lent que la moyenne mais dans un temps raisonnable et avec le même coût de la mémoire. Nous avons fixé la limite de mémoire pour le programme Magma à 500 Mo.

En fait, un habituel nécessite moins de 300 Mo de mémoire dans le cas d'un résoluble systèmes. De plus, pour le cas où le calcul de base de Grôbner nécessite plus de mémoire, pour deviner la prochaine plus grande chaîne semble être une meilleure stratégie que d'augmenter la limite de mémoire. En outre, en comparant les résultats expérimentaux pour systèmes avec un nombre différent d'équations S-box, nous concluons que pour notre attaque la variante avec $t = 23$ est plus adaptée en termes de coût en temps, par exemple, le rapport des temps d'exécution est d'environ 3,9 pour t = 39 et t = 23. Cependant, il y a des cas où la clé secrète ne peut être dérivée que si toutes les équations S-box sont inclus dans le système d'équations pour le

collisions FL sur $GF(2^8)$

Résolution d'équations pour les collisions FL

Les collisions FL conduisent, en règle générale, à des résultats plus efficaces. Chaque équation lie seulement deux variables $GF(2^8)$, puisqu'une traite des équations binomiales introduites dans la sous-section 2. Il y a 32 variables K sur $GF(2^8)$. le les relations algébriques sur ces variables sont beaucoup plus simples, car on a les deux octets de texte en clair et de texte chiffré (plus d'informations relatives aux collisions). De plus, il existe des sous-systèmes non linéaires (cycles) pouvant être résolus indépendamment (voir la sous-section precedente). En moyenne, il y a 1,02 cycles couvrant 30,08 sur 32 variables $GF(2^8)$ pour 5 mesures et 0,99 cycles couvrant 20,08 sur 32 variables $GF(2^8)$ pour 4 mesures. Statistiquement, il y a 43,58 collisions pour 5 mesures et 29,66 collisions pour 4 mesures.

Résolution des systèmes combinés

Bien que les systèmes FS et, tout d'abord, FL fonctionnent bien pour 4 et 5 mesures, leur solution pour 3 mesures est soit extrêmement improbable, soit plutôt irréalisable. Ici, une approche combinée doit être utilisée.Pour résoudre les systèmes non linéaires, nous avons exécuté l'algorithme 6. Les résultats

6 Solving combined systems of nonlinear equation

1 : if there are nonlinear cycles in the binomial chains then

2 : Resolve the cycles over $GF(2^8)$ using F4 or brute-force

3 : Define bytes of the dependent chains

4 : end if

5 : Find the h longest binomial chains

6 : Execute Algorithm 5 for sieving chain evaluations

7 : for each non-contradicting evaluation of h chains do

8 : Find Gr¨obner basis for the reduced combined system of nonlinear equations with F4

9 : if the Gr¨obner basis 6= {1} then

10 : Verify the key candidates using a known plaintext-ciphertext pair

11 : end if

12 : end for

de l'application de cet algorithme au système combiné non-linéaires des équations (avec collisions supplémentaires) pour 3 mesures peuvent être trouvées dans Tableau 10. Le système peut être résolu avec une probabilité de 0,698 dans les 22 jours ou avec une probabilité de 0,419 en 4,24 heures ou avec une probabilité de 0,072 en quelques minutes.

3.3 Synthèse et conclusion

En modélisant les attaques par collision comme des attaques ASCA, chaque chiffrement sera entièrement représenté par des systèmes d'équations et les équations de fuite seront ajoutées à ces systèmes. L'inconvénient est qu'on obtient alors des systèmes d'équations énormes puisque tout est modélisé, alors que seules les équations modélisant les collisions peuvent suffire, du moins pour les tours extrémaux. On a cependant pu modéliser des attaques ASCA avec un modèle de fuite fournissant toutes les collisions sur les tours 1, 2, 9 et 10 de l'AES et obtenir des résultats comparables avec ceux déjà publiés.[11]

Ce changement de point de vue apporte quand même la capacité de produire facilement des équations pouvant modéliser n'importe quelle collision,en particulier des collisions faisant intervenir des tours internes. Cela permettrait d'étendre ces attaques et d'outrepasser les contre-mesures qui se trouvent souvent uniquement sur les premiers et derniers tours. Nous avons donc essayé de tirer parti des collisions uniquement sur les tours internes 3 à 8 de l'AES. Les informations de fuite sont simplement rajoutées aux systèmes sous formes d'égalités entre variables intermédiaires. Malheureusement, les systèmes d'équations ainsi obtenus n'ont pas pu être résolus ni par calcul bases de Gröbner ni par des SAT solveurs, malgré un grand nombre de chiffrements différents pris en compte. Des méthodes de résolution plus adaptées ou une autre modélisation semblent nécessaires pour pouvoir étendre les attaques par collision aux tours internes. De plus le critère de réussite proposé pour les ASCA semble être assez mal adapté à ce modèle de fuite. En effet, ce critère a été défini pour s'appliquer sur un seul tour alors que les collisions sont réparties entre plusieurs tours différents.

Les attaques ASCA peuvent être étendues sous des hypothèses plus réalistes en utilisant différents modèles de fuite suivant le contexte et l'implémentation visée. On a vu que l'efficacité de l'attaque dépendait beaucoup de la quantité d'information apportée par le modèle de fuite. Par exemple, les attaques sont plus difficiles avec le modèle de la distance de Hamming qu'avec le poids de Hamming, en particulier l'étape de résolution demande bien plus de calculs. On a ansi pu adapter notre critère de réussite pour quelques autres modèles et montrer qu'il fournissait une bonne mesure pour estimer la quantité d'information qu'ils apportaient. Cela nous à aussi amené à considérer un modèle de fuite rendant les ASCA possibles en présence d'erreurs.

L'introduction de nouveaux modèles de fuite ont aussi permis de généraliser les attaques par faute et les attaques par collision, en les considérant comme des attaques ASCA. Cependant, on a pu voir que notre critère ne s'adaptait pas naturellement à ces modèles particuliers. En effet, notre critère s'appuie toujours sur une étude locale de l'influence de la fuite. Or ces modèles de fuite font interagir des opérations qui peuvent être très éloignées dans le cryptosystème. Un critère plus général prenant en compte des dépendances sur des tours différents est donc nécessaire pour mieux analyser les ASCA dans le cas général.

Bibliographie

[1] Morgan BARBIER and Pierre-Louis CAYREL. Attaque algébrique de cryptosystemes symétriques. 2008.

[2] Thomas Becker, Volker Weispfenning, and Bernd Sturmfels. Grobner bases–a computational approach to commutative algebra. *SIAM Review*, 36(2) :322–322, 1994.

[3] A Canteaut and F Lévy-dit Véhel. La cryptologie moderne. paru dans la revue armemen, 73 : 76-83. *Mars*, 2001.

[4] Stéphane Collart, Michael Kalkbrener, and Daniel Mall. Converting bases with the gröbner walk. *Journal of Symbolic Computation*, 24(3-4) :465–469, 1997.

[5] David Cox, John Little, and Donal O'Shea. Ideals, varieties, and algorithms. undergraduate texts in mathematics, 1997.

[6] Michel Dubois. *Conception, développement et analyse de systèmes de fonction booléennes décrivant les algorithmes de chiffrement et de déchiffrement de l'Advanced Encryption Standard.* PhD thesis, 2017.

[7] Jean-Charles Faugere. A new efficient algorithm for computing gröbner bases (f4). *Journal of pure and applied algebra*, 139(1-3) :61–88, 1999.

[8] Jean-Charles Faugère and Gwénolé Ars. Comparison of xl and gröbner basis algorithms over finite fields. 2004.

[9] Jean-Charles Faugere, Patrizia Gianni, Daniel Lazard, and Teo Mora. Efficient computation of zero-dimensional gröbner bases by change of ordering. *Journal of Symbolic Computation*, 16(4) :329–344, 1993.

[10] Rüdiger Gebauer and H Michael Möller. On an installation of buchberger's algorithm. *Journal of Symbolic computation*, 6(2-3) :275–286, 1988.

[11] Christopher Goyet. *Cryptanalyse algébrique par canaux auxiliaires.* PhD thesis, Université Paris 8, 2012.

[12] Martin Kreuzer and Lorenzo Robbiano. *Computational commutative algebra 2*, volume 2. Springer Science & Business Media, 2005.

[13] Daniel Lazard. Gröbner bases, gaussian elimination and resolution of systems of algebraic equations. In *European Conference on Computer Algebra*, pages 146–156. Springer, 1983.

[14] Daniel Gomes Mesquita. *Architectures reconfigurables et cryptographie : une analyse de robustesse et contremesures face aux attaques par canaux cachés.* PhD thesis, 2006.

[15] Sean Murphy and Matthew JB Robshaw. Essential algebraic structure within the aes. In *Annual International Cryptology Conference*, pages 1–16. Springer, 2002.

[16] AJM Segers. Algebraic attacks from a gröbner basis perspective. *Master's Thesis*, 2004.

[17] William Stein. Sage mathematics software. *http ://www. sagemath. org/*, 2007.