# Analyse et conception de fonctions de hachage cryptographiques

Stéphane Manuel

École Doctorale de l'X
École Polytechnique

# Analyse et conception de fonctions de hachage cryptographiques

## THÈSE

présentée et soutenue publiquement le 23 novembre 2010

pour l'obtention du titre de

**Docteur de l'École Polytechnique**

**(spécialité informatique)**

par

Stéphane Manuel

**Composition du jury**

| | | |
|---|---|---|
| *Rapporteurs :* | Pierre-Alain Fouque | (École Normale Supérieure) |
| | Antoine Joux | (Université Versaille Saint-Quentin-en-Yvelines) |
| *Directeurs :* | Daniel Augot | (INRIA École Polytechnique) |
| | Nicolas Sendrier | (CRI Paris-Rocquencourt) |
| *Examinateurs :* | Claude Carlet | (Université Paris 8) |
| | Christophe De Cannière | (K.U. Leuven) |
| | Henri Gilbert | (Agence nationale de sécurité des systèmes d'information) |

**Centre de Recherche INRIA Paris-Rocquencourt — Équipe Secret**

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

INRIA centre de recherche PARIS - ROCQUENCOURT

Mis en page avec la classe thloria.

# Remerciements

Je souhaite adresser mes premiers remerciements à Nicolas Sendrier, qui a accepté de m'accueillir tout d'abord comme stagiaire, puis comme doctorant. Je veux dire ici combien j'ai apprécié ses qualités et son ouverture d'esprit en tant que personne et en tant que chercheur. Nicolas a fait preuve d'une grande disponibilité et de beaucoup de patience pour répondre à mes questions et me guider dans le travail de recherche que j'ai mené. Je tiens aussi à remercier Daniel Augot qui a accepté de co-encadrer ma thèse avec Nicolas.

Je considère comme un privilège le fait d'avoir eu l'opportunité de pratiquer la cryptographie au sein du projet CODES où j'ai pu côtoyer nombre de personnalités parmi les plus brillantes qu'il m'ait été donné de rencontrer. Je tiens à exprimer ma reconnaissance à toute l'équipe du projet dans son ensemble, permanents, secrétaire et thésards confondus. Merci donc à Pascale, Anne, Françoise, Christelle, Marine, Marion, Andrea, Maria, Anne, Céline, Christina, Chrisanthy, Jean-Pierre, Pierre, Ayoub, Ludovic, Emmanuel, Harold, Matthieu F., Mathieu C., Cédric F., Cédric L., Yan, Frédéric, Bhaskar, Maxime, Vincent, Christophe, Stéphane, Benoît, Mamdouh, Gregory, Sumanta et Grisha.

Je voudrais aussi remercier particulièrement Claude Carlet que j'ai eu le plaisir d'avoir comme enseignant à l'université Paris 8 et grâce à qui j'ai pu découvrir la cryptographie.

Je souhaite de plus, exprimer ma gratitude à Antoine Joux et Thomas peyrin qui ont contribué au travers d'échanges, de discutions ou de collaborations à l'aboutissement d'une part importante des recherches qui sont présentées dans ce document.

Je remercie également Claude Carlet, Christophe De Cannières, Pierre-Alain Fouque, Henri Gilbert et Antoine Joux de me faire l'honneur de participer au jury de cette thèse.

Finalement, mes derniers remerciements sont pour celle qui partage ma vie depuis bientôt 17 ans, mon épouse Nathalie, qui m'a toujours soutenu dans toutes les choses que j'ai entreprises et sans qui cette thèse n'aurait probablement jamais vu le jour.

*Je dédie cette thèse à*
*Dieu,*
*ma femme,*
*mon fils et mes frères,*
*ma famille et mes amis.*

# Table des matières

Première partie

# Les fonctions de hachage cryptographiques

# 1

# Introduction

**Sommaire**



## 1-1 Principe des fonctions de hachage

Une fonction de hachage est une fonction prenant comme argument un élément de taille arbitraire finie et renvoyant un élément de longueur fixée. Une illustration du principe du hachage se trouve figure 1-1.

Ces fonctions sont très employées notamment dans le domaine des bases de données, on parle alors parfois de tables de hachage. Elles sont particulièrement utiles dans les mécanismes d'indexations qui permettent d'améliorer considérablement les performances lors de la recherche d'éléments. Cependant, les tables de hachage diffèrent fondamentalement des fonctions de hachage cryptographiques de part les propriétés que l'on attend de ces fonctions. Les fonctions de hachage cryptographiques doivent vérifier des propriétés particulières liées à leur utilisation dans le domaine de la sécurité de l'information.

**Tables de hachage.** Une table de hachage (*Hash Table*) est une structure de données permettant d'associer une valeur à une clé. Cette structure se présente le plus souvent sous la forme d'un tableau et permet d'effectuer des recherches en temps quasi constant. L'accès à une valeur se fait au moyen d'une adresse permettant de déterminer la localisation d'un élément quelconque dans le tableau. Cette adresse est obtenue en appliquant une fonction de hachage à une clé. La valeur associée à cette clé est alors stockée dans la case correspondante du tableau. La fonction de hachage transforme donc une clé de recherche en adresse permettant d'obtenir directement la valeur recherchée.

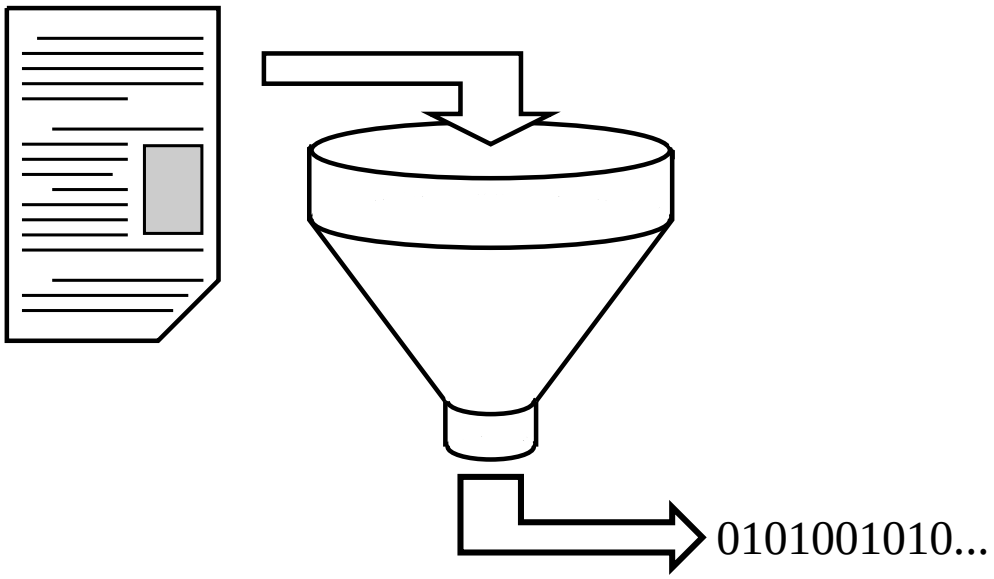



FIG. 1-1 – Principe du hachage.

Dans le cadre des tables de hachage, la fonction de hachage utilisée peut être inversible (il est possible de retrouver une clé à partir d'une adresse). De plus, l'existence de collisions (deux clés aboutissant à une même adresse), bien que demandant un traitement particulier, n'invalide pas la structure. Le fait d'autoriser ces deux comportements, différencie fondamentalement les fonctions de hachage qui sont utilisées dans les tables de hachage, des fonctions de hachage cryptographiques.

**Contrôle de redondance cyclique (CRC).** Le contrôle de redondance cyclique ou CRC (*Cyclic Redundancy Check*) est une technique permettant de détecter les erreurs pouvant advenir lors de la transmission de données. Le principe de ce contrôle consiste à ajouter aux données de la redondance. Cette redondance, communément désignée sous le nom de somme de contrôle (*Checksum*), est construite sur le principe des fonctions de hachage. Une opération rapide (par exemple un simple calcul de parité) est appliquée sur les données afin d'obtenir une empreinte. La redondance accompagne les données lors de leur transmission ou de leur stockage. Il est alors possible de recalculer la somme de contrôle, et de vérifier l'état des données. Si le nombre d'altérations est inférieur à une certaine borne, elles sont détectées.

Les contrôles de redondance cyclique possèdent l'avantage d'être très rapide à calculer. Cependant, ils ont aussi pour inconvénient de ne pas être capable, de par leur simplicité, de détecter une manipulation malicieuse des données.

**Fonctions de hachage cryptographiques.** Les fonctions de hachage cryptographiques diffèrent des autres types de fonctions de hachage par les propriétés de sécurité qui leur sont imposées. En effet, elles sont employées dans des domaines où la sécurité des données traitées est critique. Dans le cadre des tables de hachage ou des contrôles de redondance cyclique, la priorité est donnée aux performances. Une fonction de hachage cryptographique doit elle aussi être rapide à calculer. Cependant, c'est la résistance aux tentatives de manipulations malveillantes des données qui

constitue l'élément principal pris en considération lors de la conception d'une telle fonction. Les propriétés cryptographiques que l'on exige des fonctions de hachage varient selon les domaines où elles sont employées.

## 1-2 Domaines d'utilisation des fonctions de hachage cryptographiques

Les fonctions de hachage cryptographiques possèdent de nombreux domaines d'utilisation, on les qualifie parfois de "couteau suisse" de la cryptographie. Chacun de ces domaines requiert des propriétés particulières de sécurité, nous dressons dans cette section une liste non exhaustive de ces domaines.

**Intégrité des données.** Il s'agit de la fonctionnalité principale demandée à une fonction de hachage cryptographique. Elle permet de vérifier que des données n'ont pas été altérées depuis leur création ou lors de leur transmission. Le moindre changement dans les données doit, avec une très grande probabilité, aboutir à l'obtention d'empreintes différentes. Historiquement, les premières fonctions proposées pour assurer cette fonctionnalité étaient fondées sur la théorie des codes et étaient nommées codes de détection de manipulations (MDC pour *Manipulation Detection Codes*).

**Authentification de messages.** La propriété d'intégrité ne permet pas de se prémunir contre un adversaire actif qui essaierait d'altérer malicieusement les données. Un moyen de palier ce problème consiste à procéder à l'authentification de la source des données en utilisant des codes d'authentification de message (MAC pour *Message Authentication Codes*). L'objectif d'un code d'authentification de message est double : il doit permettre d'authentifier la source d'un message et de vérifier l'intégrité des données sans l'aide d'un mécanisme additionnel. L'authentification de messages relève du domaine de la cryptographie symétrique, l'utilisation d'une clé secrète étant nécessaire.

**Signature électronique.** Les schémas de signature électronique sont sans aucun doute l'application la plus importante des fonctions de hachage cryptographiques. Une signature électronique est un équivalent électronique d'une signature écrite. Elle permet de plus, de détecter si l'information signée a été altérée après sa signature. Les algorithmes utilisés pour signer des données nécessitent des calculs importants et sont donc relativement lents comparativement aux vitesses d'exécution des fonctions de hachage. Aussi, afin d'accélérer les procédures de signature et de vérification de signature, on utilise une fonction de hachage cryptographique pour calculer l'empreinte des données à signer et appliquer l'algorithme de signature à cette empreinte.

**Protection de mots de passe.** Une autre des applications courantes des fonctions de hachage cryptographiques est la protection de mots de passe. Un mot de passe est une chaîne de caractères utilisée pour authentifier l'identité d'un utilisateur ou autoriser l'accès aux ressources d'un système informatique. Il est nécessaire de protéger les mots de passe afin de les stocker. Une solution courante consiste à ne stocker que leur empreinte calculée en appliquant une fonction de hachage cryptographique à une combinaison du mot de passe et d'un sel (*Salt*).

**Dérivation de clé.** Dans le cadre de la cryptographie symétrique, les parties partagent une clé secrète commune. Il est alors fréquent que différentes clés supplémentaires soient nécessaires pour différentes applications. La dérivation de clé (ou diversification de clé) consiste à générer une ou plusieurs clés à partir d'une même valeur secrète. Cette utilisation des fonctions de hachage cryptographiques a pour but d'empêcher un adversaire ayant obtenu une clé dérivée d'obtenir des informations sur la valeur secrète ou les autres clés dérivées.

**Protocoles d'engagement.** Les fonctions de hachage cryptographiques sont aussi employées dans les protocoles d'engagement. Un protocole d'engagement consiste à permettre à une partie de s'engager sur une valeur sans divulguer aucune information sur celle-ci ; la valeur engagée étant révélée ultérieurement. Ces protocoles sont utilisés pour lier des parties à des valeurs d'engagement de façon à ce qu'aucune des parties ne puisse tirer à posteriori un avantage inapproprié sur les autres.

**Génération de nombres pseudo-aléatoires.** Un générateur pseudo-aléatoires est un algorithme déterministe qui génère une suite de bits possédant un caractère proche d'une séquence purement aléatoire. Ces algorithmes constituent, parmi d'autres utilisations, le coeur des schémas de chiffrement à flot. Les générateurs pseudo-aléatoires fondés sur une fonction de hachage cryptographique utilisent essentiellement deux modes opératoires. Le premier consiste à calculer de façon itérative à partir d'une graine (*Seed*), différentes empreintes desquelles on extrait des bits pseudo-aléatoires. Le second consiste à calculer les empreintes à partir de la graine et d'un compteur.

## 1-3 Fonctions de hachage cryptographiques

Nous introduisons dans cette section une définition algorithmique des fonctions de hachage et présentons un objet théorique dénommé oracle aléatoire qui modélise la notion de fonction de hachage cryptographique idéale. Nous présentons ensuite les trois propriétés classiques exigées des fonctions de hachage cryptographiques qui sont la résistance aux collisions, la résistance au calcul d'antécédent et la résistance au calcul de second antécédent.

### 1-3.1 Définition

Une fonction de hachage prend comme argument une chaîne de bits de longueur arbitraire finie (le message) et restitue en sortie une chaîne de bits de longueur fixée : l'empreinte (ou haché), appelée aussi parfois condensat ou simplement haché. Nous pouvons décrire une telle fonction de la façon suivante :

**Définition 1.1 (Fonction de hachage)**
*Une fonction de hachage est une fonction $h$ possédant les deux propriétés suivantes :*

1. *Compression :* $h : \{0,1\}^* \to \{0,1\}^n$.
2. *Facilité de calcul :* $h$ *et* $x \in \{0,1\}^*$ *donnés, on peut calculer efficacement* $h(x)$.

La notation $\{0,1\}^*$ désigne l'ensemble des chaînes de bits de longueur arbitraire finie et la notation $\{0,1\}^n$ désigne l'ensemble des chaînes de bits de longueur exactement $n$. Cette définition prend en compte les deux aspects fondamentaux exigés de toute fonction de hachage (cryptographique ou non) : elle doit permettre d'une part d'obtenir une empreinte de taille réduite et d'autre part être rapide à calculer. Il s'agit d'une définition des fonctions de hachage selon un point de vue algorithmique.

Bellare et Rogaway [BR93] ont introduit en 1993 un objet théorique dénommé oracle aléatoire (*Random Oracle*). Un oracle aléatoire est un objet de type boîte noire, qui répond à chaque requête issue de son ensemble de départ par une réponse aléatoire choisie uniformément dans son ensemble d'arrivée. Pour un ensemble *Dom* et un ensemble fini *Rng* un oracle aléatoire est défini par une machine de Turing acceptant des entrées $X \in Dom$ :

**Algorithm 1** Oracle aléatoire $RO_{Dom,Rng}(X)$

**if** $T[X] = \perp$ **then**
  $T[X] \xleftarrow{\$} Rng$
**end if**
**return** $T[X]$
où $T$ est une table intégralement initialisée à $\perp$.

Un oracle aléatoire est dit consistant, s'il donne systématiquement une réponse identique pour une même question posée. De plus, on distingue usuellement les oracles aléatoires à entrée de longueur fixe (*Fixed Input Length*) qui correspondent à une fonction de compression[1] idéale, et les oracles aléatoires à entrée de longueur variable (*Variable Input Length*) qui correspondent à une fonction de hachage idéale.

Un oracle aléatoire constitue une fonction de hachage cryptographique parfaite. Il vérifie toutes les propriétés exigées des fonctions de hachage cryptographiques. Il s'agit d'une fonction aléatoire pour laquelle il n'existe pas d'attaques strictement meilleures que les attaques génériques.

### 1-3.2 Propriétés classiques

La spécificité d'une fonction de hachage cryptographique réside, comme nous l'avons déjà souligné, dans ses propriétés de sécurité supplémentaires. Nous introduisons ici les trois propriétés, dites classiques, exigées pour ce type de fonction.

En pratique, les fonctions de hachage bornent par construction la taille maximale des messages susceptibles d'être traités. Pour les fonctions SHA-0 et SHA-1 par exemple, la taille maximale d'un message est égale à $2^{64}-1$ bits. Pour les définitions des propriétés cryptographiques que nous allons présenter, nous choisissons de restreindre l'ensemble de définition de la fonction $h$ ce qui permet de définir une distribution de probabilité uniforme sur l'ensemble des messages. Soit $r$ un entier naturel, nous considérons donc la fonction de hachage cryptographique $h : \{0,1\}^r \longrightarrow \{0,1\}^n$.

**Résistance aux collisions (*Collision Resistance*).** La première propriété que l'on exige d'une fonction de hachage cryptographique est d'être résistante aux colli-

[1] Nous développerons les fonctions de compression dans le chapitre consacré à la construction des fonctions de hachage.

sions. On peut définir cette propriété de la façon suivante :

**Propriété 1.1 (Résistance aux collisions)**
*La fonction $h$ résiste de façon optimale aux collisions si on ne possède pas d'algorithme capable de produire un couple de messages $(x, x') \in \{0,1\}^r \times \{0,1\}^r$, tel que $x' \neq x$ et $h(x') = h(x)$ avec une complexité meilleure que $\mathcal{O}(2^{n/2})$ opérations. Un tel couple forme une collision pour la fonction de hachage $h$.*

La résistance aux collisions correspond à la propriété de sécurité spécifique à l'intégrité des données. Cependant pour une fonction de hachage cryptographique, le fait de ne pas résister de façon optimale aux collisions constitue une faiblesse. On considère donc, qu'une fonction pour laquelle cette propriété est mise en défaut est inapte à être utilisée dans l'ensemble des domaines cryptographiques.

**Résistance au calcul d'antécédent (*Preimage Resistance*).** La résistance au calcul d'antécédent illustre le caractère non-inversible d'une fonction de hachage :

**Propriété 1.2 (Résistance au calcul d'antécédent)**
*Soit un message $m$ tiré aléatoirement dans $\{0,1\}^r$ (ensemble de définition de la fonction $h$) on pose $y = h(m)$. La fonction $h$ résiste de façon optimale au calcul d'antécédent si on ne possède pas d'algorithme capable de produire un antécédent $x \in \{0,1\}^r$ tel que $h(x) = y$, avec une complexité meilleure que $\mathcal{O}(2^n)$ opérations.*

La résistance au calcul d'antécédent telle que nous la définissons diffère de la résistance à l'inversion. En effet, l'attaquant n'a pas l'obligation de produire le message $m$ utilisé pour générer $y$, mais un message quelconque possédant la même empreinte. Cette propriété est particulièrement désirable dans le contexte de la protection de mot de passe et de la dérivation de clé.

**Résistance au calcul de second antécédent (*Second Preimage Resistance*).** La propriété de résistance au calcul de second antécédent est particulièrement requise dans les schémas de signature électronique. On peut la définir de la façon suivante :

**Propriété 1.3 (Résistance au calcul de second antécédent)**
*Soit un message $x$ tiré aléatoirement dans $\{0,1\}^r$. La fonction $h$ résiste de façon optimale au calcul de second antécédent si on ne possède pas d'algorithme capable de produire un second antécédent $x' \in \{0,1\}^r$ tel que $x' \neq x$ et $h(x') = h(x)$, avec une complexité meilleure que $\mathcal{O}(2^n)$ opérations.*

La définition de la résistance au calcul de second antécédent peut sembler similaire à celle de la résistance aux collisions. Cependant, ces deux propriétés diffèrent sur un point essentiel : le degré de liberté accordé à l'adversaire. Pour la résistance aux collisions, l'adversaire est libre du choix des messages $x$ et $x'$. Dans le cadre de la résistance au calcul de second antécédent, le message $x$ et donc le haché $y$ sont imposés.

Une illustration des modèles d'attaques correspondant à ces propriétés est proposé figure 1-2.

Ces trois définitions forment le corpus de propriétés minimales que doit posséder toute fonction de hachage afin de pouvoir être qualifiée de cryptographique. Il existe de nombreuses autres propriétés en plus de ces trois propriétés classiques. Certains

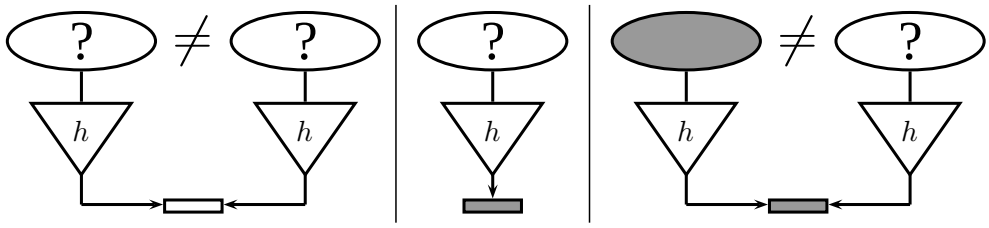



FIG. 1-2 – Cette figure illustre les différents objectifs d'un adversaire souhaitant mettre en défaut les propriétés de résistance aux collisions, de résistance au calcul d'antécédent et de résistance au calcul de second antécédent. Les parties grisées correspondent aux éléments qui sont imposés et les points d'interrogation figurent les messages que doit produire l'adversaire.

domaines d'application nécessitent que la fonction utilisée soit de plus indistinguable d'une fonction aléatoire. C'est le cas par exemple pour la dérivation de clé et la génération de nombre pseudo-aléatoires. On parlera alors de fonction pseudo-aléatoire (*Pseudo Random Function*) pour caractériser le fait que la sortie de cette fonction ne peut être distinguée de celle d'une fonction purement aléatoire. Le tableau1-1 établit une correspondance entre ces propriétés et certains domaines d'utilisation des fonctions de hachage cryptographiques introduits précédemment.

| Domaines | Propriétés | | | |
|---|---|---|---|---|
| | *Col* | *Pre* | *Sec* | *PRF* |
| Intégrité des données | × | | | |
| Signature électronique | | | × | |
| Protection de mot de passe | | × | | |
| Dérivation de clé | | × | | × |
| Génération de nombres pseudo-aléatoires | | × | | × |

TAB. 1-1 – Tableau de correspondance entre les propriétés des fonctions de hachage cryptographiques et les domaines d'utilisation. Les notations *Col*, *Pre*, *Sec* et *PRF* désignent respectivement la résistance aux collisions, au calcul d'antécédent, au calcul de second antécédent et le caractère pseudo-aléatoire. La présence d'une croix indique le fait qu'un adversaire mettant en défaut la propriété correspondante peut être directement utilisé pour invalider l'utilisation de la fonction dans le domaine. Les domaines de l'authentification de message et des protocoles d'engagement ne sont pas présent dans ce tableau car les propriétés qui leur correspondent varient en fonction des constructions utilisées.

**Terminologies alternatives.** On peut trouver dans la littérature consacrée des terminologies alternatives pour ces propriétés :

– sens unique (*One Way*) pour résistance au calcul d'antécédent,

- faible résistance aux collisions (*Weak Collision Resistance*) pour résistance au calcul de second antécédent,
- forte résistance aux collisions (*Strong Collision Resistance*) pour résistance aux collisions.

On nomme fonction de hachage à sens unique (*One Way Hash Function*) ou fonction de hachage à sens unique faible (*Weak One Way Hash Function*), une fonction de hachage satisfaisant les propriétés 1.1 et 1.2. On nomme fonction de hachage résistante aux collisions (*Collision Resistant Hash Function*) ou fonction de hachage à sens unique forte (*Strong One Way Hash Function*), une fonction de hachage satisfaisant les propriétés 1.1 et 1.3.

**Définitions formelles.** Les propriétés que nous avons énoncées ne constituent pas des définitions d'un point de vue rigoureusement formel. Dans ces propriétés, nous utilisons la notion de résistance optimale. En effet, nous pouvons associer à chacune de ces trois propriétés une attaque générique. Notons $n$ la taille, exprimée en nombre de bits, de la sortie d'une fonction de hachage. Pour la résistance aux collisions, le paradoxe des anniversaires montre que l'on peut trouver une collision avec une complexité de l'ordre de $\mathcal{O}(2^{n/2})$ appels à la fonction. En ce qui concerne la résistance au calcul d'antécédent et la résistance au calcul de second antécédent, une recherche exhaustive de complexité $\mathcal{O}(2^n)$ appels à la fonction permet d'obtenir l'objet désiré. Ces deux attaques, que nous détaillerons section 2-5, sont dites génériques car elles fonctionnent quelle que soit la fonction de hachage considérée. Une fonction résiste alors de façon optimale, et est donc qualifiée de résistante, si aucune attaque strictement meilleure que l'attaque générique correspondante n'est connue.

Un traitement théorique rigoureux des propriétés cryptographiques des fonctions de hachage a été considéré pendant de nombreuses années comme étant vain. En effet si l'on considère la résistance aux collisions, définir de façon rigoureuse cette propriété en utilisant le formalisme classique ne fonctionne tout simplement pas. Il est évident qu'une multitude de collisions existent. En effet par définition, le cardinal de l'ensemble de départ d'une fonction de hachage est bien plus grand que le cardinal de son ensemble d'arrivée. Il en résulte qu'un grand nombre d'éléments possèdent une même image. Il existe donc un algorithme trivial capable d'invalider cette propriété. Cet algorithme possède une complexité en temps constant, ne nécessite aucune requête et possède une probabilité de succès égale à 1. Il s'agit de l'algorithme qui possède l'une des collisions existantes codée "en dur", et se contente de la restituer en sortie.

Afin de palier à ce problème on utilise la notion de famille de fonctions de hachage cryptographiques qui permet de définir formellement les propriétés de sécurité classiques et d'en introduire de nouvelles à même de couvrir l'éventail des utilisations de ces fonctions. Des études détaillées de ces propriétés, ainsi que des liens qui les unissent, ont été menées dans un article de Rogaway et Shrimpton [RS04] et plus récemment dans un article de Reyhanitabar *et al.* [RSY10].

## 1-4 Bref historique des fonctions de hachage cryptographiques

Le besoin de fonctions de hachage cryptographiques est tout d'abord apparu dans le contexte de l'authentification relativement à la protection de mots de passe. On désignait alors ces fonctions sous le terme de fonctions à sens unique (*One-Way Functions*), afin d'indiquer que ces fonctions devait être résistantes à l'inversion. Whitfield Diffie et Martin E. Hellman [DH76] ont été les premiers à définir une telle fonction à sens unique. Les notions de résistance au calcul de second antécédent et de résistance aux collisions ont été développées consécutivement. Diffie et Hellman ont aussi montré dans le même article, comment un algorithme de chiffrement sûr pouvait être utilisé pour créer une fonction de hachage. Au début des années 1980, les premiers schémas concrets ont été proposés : les constructions de Davies-Meyer et Matyas-Meyer-Oseas employées avec l'algorithme de chiffrement par bloc DES (*Data Encryption Standard*) [FIPS-DES]. Cependant, du fait de la taille réduite des blocs du DES (64 bits), la sécurité de ces constructions était insuffisante. Les constructions MDC-2 [MS88] proposée en 1988 puis MDC-4 [CPM90] en 1990 constituent les premières fonctions fondées sur le DES munies d'un niveau de sécurité permettant leur utilisation en pratique. Durant cette même période, une des première fonction de hachage ad hoc MD2 [RFC-MD2] était développée par Ronald L. Rivest. MD2 fût remplacé par MD4 [RFC-MD4] en 1990 puis par MD5 [RFC-MD5] en 1992. La fonction MD5 est la première fonction de hachage cryptographique à avoir fait l'objet d'une large utilisation. En 1993, l'institut national de standardisation américain (*Nationnal Institute of Standards and Technology* NIST) normalisât la fonction de hachage SHA (*Standard Hash Algorithm*), construite selon les idées et principes de MD4. La fonction SHA, rebaptisée SHA-0 [FIPS-SHA0], fût retirée en 1995 selon les instructions de la NSA (*Nationnal Security Agency*) au profit du nouveau standard SHA-1 [FIPS-SHA1]. En 2002, une nouvelle fonction de hachage cryptographiques SHA-2 [FIPS-SHA2a], puis en 2004 ses différentes variantes [FIPS-SHA2b] ont été normalisées par le NIST.

Les fonctions les plus employées en pratique sont MD5 et SHA-1. Mais ces fonctions ont fait l'objet ces dernières années de nombreuses cryptanalyses efficaces qui dans le cas de MD5 se sont révélées dévastatrices. Le standard actuel SHA-2 est encore aujourd'hui considéré comme sûr. Cependant, le modèle de conception employé pour les différentes versions de SHA-2 est relativement proche de celui utilisé par les standards précédents ; on parle des fonctions de la famille MD-SHA (*MD-SHA Familly*) pour désigner les différentes fonctions MD4, MD5, SHA-0, SHA-1 et SHA-2. Aussi, le NIST a ouvert une compétition publique pour développer un nouvel algorithme de hachage cryptographique destiné à être employé dans les schémas de signature électronique, l'authentification de message et autres applications cryptographiques. Cette compétition, destinée à répondre aux récents développements des cryptanalyses des fonctions de hachage, c'est ouverte le 31 octobre 2008. Le nouvel algorithme, baptisé sans surprise SHA-3 [FIPS-SHA3], devrait être choisi et validé à la fin de l'année 2012.

# 2

# Constructions classiques et leur sécurité

**Sommaire**



## 2-1 Introduction

Nous avons introduit les fonctions de hachage comme étant des fonctions prenant comme argument un élément de taille arbitraire finie et renvoyant un élément de longueur fixée. Construire une telle fonction, vérifiant les propriétés de sécurité que nous avons énoncées, n'est pas une chose aisée. Une solution naturelle pour résoudre ce problème consiste à combiner une primitive opérant sur un domaine de taille fixe et un algorithme capable d'étendre le domaine de cette primitive. Les premières propositions fondées sur ce paradigme ont été présentées en 1978 par Rabin [Rab78] puis en 1979 par Merkle [Mer79a, Mer79b]. En 1992, Lai et Massey [LM92] proposèrent un algorithme qu'ils baptisèrent fonction de hachage itérative. Une illustration du principe du hachage itératif se trouve figure 2-1.

Les constructions itératives utilisent une fonction de compression combinée à un algorithme d'extension de domaine. La fonction de compression est souvent l'élément le plus vulnérable des fonctions de hachage cryptographiques itératives et donc

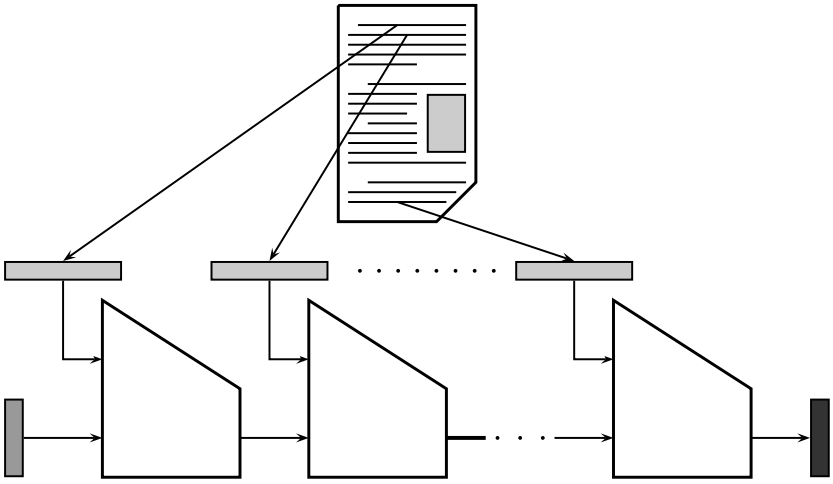


FIG. 2-1 – Hachage itératif. Le message est découpé en éléments de taille fixe qui sont traités itérativement.

le plus difficile à concevoir. La principale difficulté repose sur le compromis nécessaire entre vitesse d'exécution et sécurité. Une fois la fonction de compression construite, on applique l'extenseur de domaine qui doit lui-même ne pas introduire de faiblesse supplémentaire. L'algorithme d'extension de domaine le plus populaire est certainement l'algorithme de Merkle-Damgård [Mer89, Dam89]. Cependant, avec l'intensification de la recherche sur les fonctions de hachage cryptographiques, de nouvelles propositions d'extenseurs de domaine ont vu le jour, parmi lesquelles on peut citer les éponges cryptographiques [BDP06]. Bien qu'elles soient les plus utilisées en pratique, les fonctions itératives ne constituent pas la seule option possible. D'autres principes de constructions ont été proposés dont les arbres de Merkle [Mer79b].

Une attaque contre une fonction de hachage cryptographique est un algorithme dont le but est de violer une des propriétés de sécurité revendiquée par cette fonction. Dans le monde de la cryptologie, il existe différentes notions pour définir les attaques. Une façon de distinguer les types d'attaque consiste à considérer la quantité d'information disponible pour un adversaire. Selon ce point de vue, nous pouvons diviser les attaques en deux catégories. Les attaques génériques traitent la fonction de hachage comme une boîte noire (*Black Box*), et sont indépendantes du type de construction sur lequel repose la fonction de hachage. Par opposition, les attaques spécifiques considèrent la fonction de hachage comme une boîte blanche (*White Box*). Les attaques spécifiques tirent partie de structures particulières à l'algorithme utilisé pour construire la fonction de hachage. Plusieurs attaques spécifiques relatives à l'utilisation d'un extenseur de domaine et plus particulièrement à l'algorithme de Merkle-Damgård ont été mises en évidence. Ces attaques permettent de distinguer trivialement une fonction de hachage itérative d'une fonction de hachage parfaite (oracle aléatoire). La notion d'indifférentiabilité introduite par Maurer *et al.* [MRH04] en 2004 permet de prendre en compte l'impact de l'extenseur de domaine sur la sécurité d'une fonction de hachage.

Dans ce chapitre, nous commencerons par présenter les différents principes utilisés pour la constructions de fonctions de compression avant de détailler l'algorithme

de Merkle-Damgård et de présenter deux autres algorithmes d'extension de domaine. Nous introduirons après cela les attaques génériques relatives aux propriétés de résistance aux collisions, au calcul d'antécédent et au calcul de second antécédent. Ces attaques constituent les références permettant d'établir les niveaux de sécurité d'une fonction de hachage relativement à ces propriétés. Nous détaillerons ensuite les attaques spécifiques particulières aux fonctions de hachage itératives et clorons ce chapitre en présentant notion d'indifférentiabilité.

## 2-2 Fonction de compression

La construction d'une fonction de compression est en elle-même un challenge. C'est sur cette fonction que résident les performances d'une fonction de hachage itérative et une part importante de sa sécurité (avec l'algorithme d'extension de domaine). Malheureusement, il s'avère particulièrement difficile de concilier ces deux impératifs. On peut cataloguer les différents types de conception de fonctions de compression en deux familles. La première de ces famille regroupe les constructions fondées sur des algorithmes de chiffrement par blocs. Au sein de cette famille on peut encore distinguer différentes catégories selon que l'algorithme de chiffrement utilisé est normalisé ou s'il s'agit d'un algorithme ad hoc. La seconde famille regroupe les fonctions mettant en oeuvre des algorithmes dont la sécurité est réductible à la difficulté de résoudre des problèmes algorithmiques réputés durs.

### 2-2.1 Fonctions de compression fondées sur un algorithme de chiffrement par bloc

Le principe des constructions fondées sur un algorithme de chiffrement par bloc consiste à s'appuyer sur une primitive cryptographique existante pour laquelle on possède de bons arguments de sécurité. C'est le cas, par exemple, des schémas de chiffrement par bloc. Ces fonctions de compressions présentent l'avantage de permettre de faire reposer la sécurité relative à une ou plusieurs des propriétés exigées des fonctions de hachage cryptographiques sur la sécurité de l'algorithme de chiffrement mis en oeuvre.

On distingue les constructions fondées sur des algorithmes de chiffrement par bloc selon leur taux de hachage. La notion de taux de hachage a été introduite en 1993 par Preneel [Pre93]. Puis en 1994, Knudsen et Lai [KL94], ont défini le taux de hachage comme le ratio du nombre de blocs de message traités par le nombre d'appels à l'algorithme de chiffrement employé. Cette définition constitue la mesure standard utilisée pour évaluer l'efficacité des fonctions de hachage fondées sur un schéma de chiffrement par bloc.

Les constructions de ratio 1/1 on fait l'objet d'études intensives. Parmi ces constructions, les modes opératoires les plus populaires sont Davies-Meyer, Matyas-Meyer-Oseas et Miyaguchi-Preneel. Ces trois constructions sont illustrées figure 2-2.

Preneel *et al.* [PGV93] puis Black *et al.* [BRS02] ont analysé les 64 constructions différentes possibles. Parmi elles, 12 constructions se sont avérées êtres sûres : des réductions de sécurité dans le modèle du schéma de chiffrement par bloc idéal (*Ideal Block Cipher Model*) ont été démontrées. Cependant, l'inconvénient majeur des constructions de ratio 1/1 réside dans la taille de sortie des algorithmes de chiffrement par bloc susceptibles d'être utilisés. En effet, le candidat idéal pour une telle

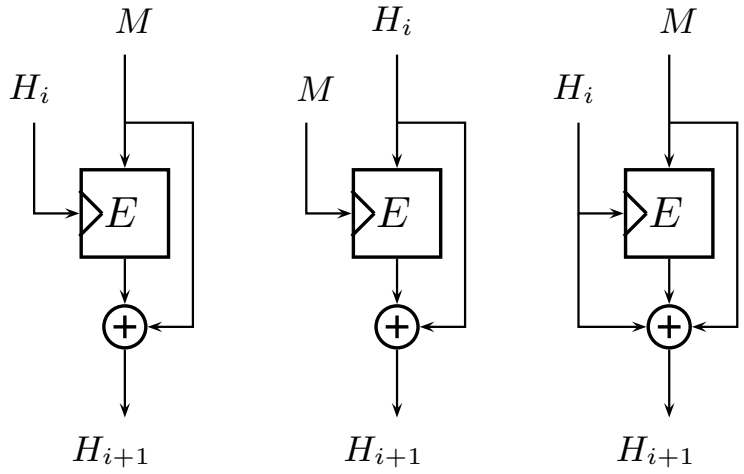



FIG. 2-2 – Modes opératoires de Davies-Meyer, Matyas-Meyer-Oseas et Miyaguchi-Preneel.

construction est l'AES [FIPS-AES] (*Advanced Encryption Standard*) qui a succédé au DES [FIPS-DES] (*Data Encryption Standard*) en 2001. La taille des blocs utilisés par l'AES est de 128 bits (64 pour le DES). Cette taille se révèle insuffisante pour résister à l'attaque générique par collision au vu des critères de sécurité actuels. Ces critères imposent aux fonctions de hachage cryptographiques des tailles d'empreinte de 224/256 bits en architecture 32 bits et 384/512 bits en architecture 64 bits (selon le récent appel d'offre du NIST).

Afin de palier à cet inconvénient, différentes stratégies ont été explorées.

**Algorithmes de chiffrement ad hoc.** La première stratégie consiste à construire des algorithmes de chiffrement par bloc ad hoc utilisant des tailles de bloc plus grandes. Les fonctions de compression dénommées ad hoc par extension, sont employées dans la plupart des fonctions cryptographiques utilisées en pratique. Le choix de ce type de construction réside essentiellement dans le gain de performance qu'il apporte. Cependant, la sécurité de ces fonctions repose uniquement sur l'incapacité de la communauté cryptographique à produire des cryptanalyses et repose en général sur un consensus plutôt que sur une preuve. La plupart des standards adoptés par l'industrie reposent sur ce type de fonction. On peut citer les fonctions de la famille MD-SHA, qui regroupe les fonctions MD [RFC-MD4, RFC-MD5, PB95] et les fonctions SHA [FIPS-SHA0, FIPS-SHA1, FIPS-SHA2a, FIPS-SHA2b]. Les fonctions SHA-0 et SHA-1 seront décrites en détail dans la deuxième partie de cette thèse. Rijmen et Barreto ont proposé en 2000 une nouvelle fonction de hachage cryptographique, Whirlpool [BR00], dont la fonction de compression repose sur un algorithme de chiffrement utilisant des blocs de taille 512 bits. Cependant cet algorithme de chiffrement, bien que fondé sur les mêmes principes que l'AES, n'a pas fait l'objet d'autant d'attention de la part des cryptanalystes. Par conséquent, il n'y a pas de consensus établi sur sa sécurité. De plus, les performances de Whirlpool sont inférieures à la plupart des fonctions précédentes. Ce qui explique qu'elle n'ait pas été plébiscitée par l'industrie, bien qu'ayant été normalisée en 2004, sous la désignation

ISO/IEC 10118-3.

Comme nous l'avons souligné, il est difficile d'évaluer la sécurité des fonctions de compression ad hoc. Jusqu'en 2005 et bien qu'encore très employée dans l'industrie, la fonction MD5 avait déjà fait l'objet de différentes attaques [Ber92, DBB91, DBB93, Dob96a, Dob96b]. Mais le standard SHA-1 était encore considéré comme sûr. Les cryptanalyses menées contre sa version préliminaire SHA-0 [CJ98, BC04] ne semblant pas être directement transposables à la version normalisée. Cependant, l'introduction de nouvelles méthodes de cryptanalyses initiée par Wang *et al.* [WFL04, WLF05, WYY05a, WYY05b, WYY05c, WYY05d, WY05] devaient sonner le glas de pans entiers des fonctions de hachage de la famille MD-SHA. Une part importante des travaux de recherche présentés dans cette thèse concernant les fonctions SHA-0 et SHA-1 s'appuient sur ces nouvelles méthodes de cryptanalyse. Ces méthodes se sont montrées si efficaces que le NIST a décidé de lancer une compétition pour définir un nouveau standard de hachage cryptographique. On peut remarquer que malgré cet incident, les fonctions ad hoc sont toujours très populaires.

**Constructions de ratios inférieurs.** Une autre approche pour augmenter la taille des blocs consiste à définir des constructions de ratios inférieurs, utilisant plusieurs instances du chiffrement par bloc au sein de la fonction de compression. On peut alors reformuler la question de la taille de l'empreinte sous une forme différente : comment produire des empreintes de taille multiple de la taille du chiffrement par bloc utilisé ? Les premières réponses à cette question ont été les constructions MDC-2 et MDC-4 [CPM90, MS88] qui furent bientôt suivies par de nombreuses autres propositions. Cependant, peu de ces propositions ont résisté aux cryptanalyses. Certains auteurs se sont appuyés sur des arguments de théorie des codes pour donner des arguments de sécurité pour leurs constructions [KP96, KP97, KP02]. Mais là encore un problème demeure, aucune de ces constructions n'est idéale : pour une empreinte de taille $2n$ bits elles utilisent plus de 2 appels à un chiffrement par bloc de taille $n$ bits. En 2006, Peyrin *et al.* [PGM06] ont conjecturé que pour construire une fonction de compression produisant des sorties de taille 256 bits, au moins cinq appels à la fonction AES sont nécessaires pour traiter un bloc de message de taille 128 bits. Toutes les pistes relatives à cette question n'ont pas encore été épuisées, ce sujet de recherche reste ouvert.

### 2-2.2 Fonctions de compression fondées sur un problème réputé difficile

L'idée des fonctions de compression fondées sur des problèmes réputés difficiles est de faire reposer la sécurité de la fonction sur une conjecture éprouvée. En effet, nous connaissons un certains nombre de problèmes pour lesquels aucun algorithme de résolution efficace (en terme de complexité algorithmique) n'est connu. Parmi ces problèmes on peut citer, entre autres, le problème SAT, le problème de factorisation ou le décodage de syndrome. Ces problèmes ont fait l'objet d'études intensives depuis de nombreuses années et, bien que l'on ne puisse pas formellement prouver qu'ils soient impossibles à résoudre à l'aide d'algorithmes de complexité polynomiale, il est couramment conjecturé et admis que c'est effectivement le cas. L'intérêt d'utiliser ces problèmes comme base pour des fonctions de compression réside dans le fait qu'il est peu vraisemblable que des algorithmes efficaces capables de résoudre ces

problèmes puissent voir le jour dans un avenir proche. Cela met théoriquement ces fonctions à l'abri d'attaques dévastatrices telles qu'ont pu les connaître certaines des constructions ad hoc les plus populaires.

Des fonctions de hachage cryptographiques dont la sécurité repose sur de tels problèmes ont été proposées [BM97, CJ98, BAC08, AFG08, ADL08]. Paradoxalement, certaines d'entre elles ont subit des attaques peu après leur publication. En effet, les fonctions fondées sur ces problèmes sont en général peu performantes par rapport aux fonctions ad hoc, ce qui a pu conduire certains concepteurs à proposer des paramètres sous-dimensionnés. De plus, la définition des problèmes réputés difficiles induit souvent qu'ils le sont pour une instance du problème tirée aléatoirement. On parle aussi parfois de difficulté en moyenne, ce qui désigne le fait que la plupart des instances d'un problème sont difficiles à résoudre. Cependant il peut aussi exister des instances faciles même pour un problème difficile. Il est malencontreusement arrivé que certaines fonctions de hachage proposées utilisaient des paramètres constituant une instance facile d'un problème. Enfin, la construction choisie pour mettre en oeuvre le problème algorithmique peut avoir des conséquences sur la qualité de la réduction de sécurité. Cependant, les quelques échecs de fonctions de hachage fondées sur des problèmes réputés difficiles ne remettent pas en cause la validité du principe.

## 2-3 Extenseur de domaine

Une fonction de compression permet d'obtenir une empreinte pour des messages de taille fixée. Une fonction de hachage doit de permettre d'obtenir une empreinte pour des messages de taille arbitraire finie. Les constructions itératives utilisent un algorithme d'extension du domaine de la fonction de compression pour atteindre cet objectif. Cependant, l'algorithme utilisé ne doit pas introduire de nouvelle vulnérabilité et devrait idéalement préserver les propriétés cryptographiques de la fonction de compression.

L'algorithme d'extension de domaine le plus utilisé en pratique est l'algorithme de Merkle-Damgård. Cependant, de nouvelles constructions ont vu le jour durant ces dernières années. Parmi celles-ci, les éponges cryptographiques proposées par Bertoni *et al.* [BDP06] on fait l'objet d'un certain intérêt de la part de la communauté.

### 2-3.1 Algorithme de Merkle-Damgård

En 1989, Merkle [Mer89] et Damgård [Dam89] développèrent de façon indépendante un algorithme d'extension de domaine préservant certaines des propriétés cryptographiques de la fonction de compression utilisée. Une illustration graphique de l'algorithme de Merkle-Damgård se trouve figure 2-3.

Le principe de ce que l'on nomme la méta-méthode de Merkle-Damgård est décrit par l'Algorithme 2.

La première étape de l'algorithme se nomme rembourrage (*Padding*). Ce rembourrage remplit deux objectifs :

- il permet de ramener la taille du message à hacher à un multiple de $r$,
- le codage de la longueur exacte du message dans un bloc supplémentaire assure, de plus, qu'aucun message ne peut constituer le préfixe d'un autre message.

De façon générale, on parle de rembourrage sans préfixe (*Prefix Free Padding*) pour désigner toute méthode de rembourrage remplissant ces deux objectifs.

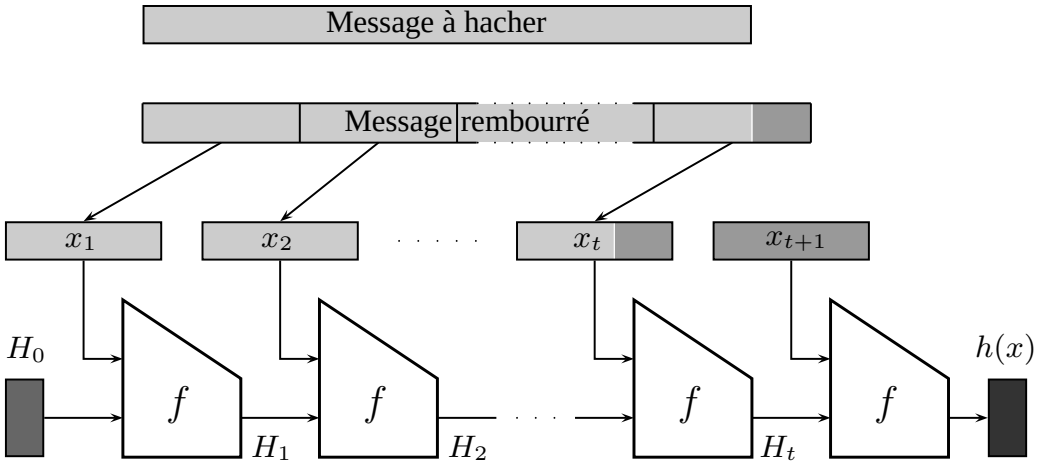



FIG. 2-3 – Algorithme de Merkle-Damgard.

**Algorithm 2** Méta-méthode de Merkle-Damgård

ENTRÉE : une fonction de compression $f : \{0,1\}^n \times \{0,1\}^r \longrightarrow \{0,1\}^n$ résistante aux collisions.
SORTIE : une fonction de hachage $h : \{0,1\}^* \longrightarrow \{0,1\}^n$ résistante aux collisions.

Découper un message $x$ de taille $b$ bits en blocs $x_1, x_2, \ldots, x_t$ de taille $r$ bits en ajoutant des bits 0 au bloc $x_t$ si nécessaire, et définir un bloc supplémentaire $x_{t+1}$, destiné à contenir le codage de la longueur du message.

$H_0 = 0^n$ $\{n$ *bits à 0*$\}$
$i = 1$
**while** $i \leq t + 1$ **do**
  $H_i = f(H_{i-1} \parallel x_i)$
**end while**
L'empreinte du message $x$ est alors $h(x) = H_{t+1} = f(H_t \parallel x_{t+1})$.

Les variables $H_i$ sont appelées variables de chaînage, et la variable $H_0$ est qualifiée de valeur initiale (*Initial Value*).

En pratique, l'algorithme de Merkle-Damgård se présente sous une forme légèrement différente. Le rembourrage est réalisé en ajoutant un bit à 1 suivi d'autant de bits à 0 que nécessaires afin de permettre l'insertion du codage de la longueur du message sur les dernier bits d'un bloc complet. Le nombre de bits $b$ réservé au codage de la longueur détermine la taille maximale des messages ($2^b$ bits). De plus, la valeur initiale est fixée lors de la spécification de la fonction de hachage.

**Préservation de la résistance aux collisions.** La popularité de l'algorithme de Merkle-Damgård découle de sa capacité à préserver la résistance aux collisions de la fonction de compression. Afin de démontrer formellement cette propriété nous introduisons une nouvelle définition des fonctions de hachage sous la forme d'une famille paramétrée par une clé. Cette étape est nécessaire afin de pouvoir donner une définition formelle de la résistance aux collisions et démontrer rigoureusement le théorème de Merkle-Damgård.

On définit une famille de fonctions de hachage par une famille de fonctions

$$\mathcal{H} : \mathcal{K} \times \mathcal{D} \to \mathcal{R},$$

où $\mathcal{K}$ est l'ensemble des clés, $\mathcal{D}$ est l'ensemble de définition et $\mathcal{R}$ l'ensemble d'arrivée. Pour une clé particulière $k \in \mathcal{K}$, la fonction $h_K : \mathcal{D} \to \mathcal{R}$ est définie pour tout $M \in \mathcal{D}$ par $h_k(M) = h(k, M)$. Il s'agit de l'instance de $h$ caractérisée par la clé $k$.

**Définition 2.1 (Famille de fonctions de hachage)**
*Une famille de fonctions de hachage $\mathcal{H}$ est constituée d'une paire d'algorithmes polynomiaux $(\Theta, h)$ satisfaisant les propriétés suivantes :*

1. *$\Theta$ est un algorithme probabiliste prenant en entrée un paramètre de sécurité $n$ et produisant en sortie une clé $k$.*
2. *Il existe un polynôme $\ell$ tel que $h$ prend en entrée une clé $k$ et une chaîne de bits $x \in \{0,1\}^*$ et produit en sortie une chaîne de bits $h_k(x) \in \{0,1\}^{\ell(n)}$.*

*Si $h_k$ est définie seulement pour des entrées $x \in \{0,1\}^{\ell'(n)}$ et $\ell'(n) > \ell(n)$, alors nous dirons que $(\Theta, h)$ est une famille de fonctions de compression prenant en entrée des chaînes de bits de taille $\ell'(n)$.*

Pour une famille de fonctions de hachage $\mathcal{H} = (\Theta, h)$, un adversaire $\mathcal{A}$ et un paramètre de sécurité $n$, nous définissons l'expérience suivante :

**Algorithm 3** Expérience de recherche de collision $Coll_{\mathcal{A},\mathcal{H}}(n)$

1. Une clé $k$ est générée à partir de l'algorithme $\Theta(n)$.
2. Étant donnée $k$, l'adversaire $\mathcal{A}$ produit deux messages $x$ et $x'$ (dans le cas d'une fonction de compression, on demande de plus que $x, x' \in \{0,1\}^{\ell'(n)}$)
3. La sortie de l'expérience est définie comme étant égale à 1 si et seulement si $x \neq x'$ et $h_k(x) = h_k(x')$. Dans ce cas nous disons que $\mathcal{A}$ a produit une collision pour l'instance $h_k$.

Nous définissons à présent la résistance aux collisions de la façon suivante :

**Propriété 2.1 (Résistance aux collisions (famille))**
*Une famille de fonctions de hachage $\mathcal{H} = (\Theta, h)$ est résistante aux collisions si pour*

*tout adversaire probabiliste $\mathcal{A}$ polynomial en temps, il existe une fonction négligeable $g$ telle que*

$$Pr[Coll_{\mathcal{A},\mathcal{H}}(n) = 1] \leq g(n)$$

Nous pouvons dès lors énoncer le théorème de Merkle-Damgård :

**Théorème 2.1 (Théorème de Merkle-Damgård)**
*Soit une famille de fonctions de hachage $\mathcal{H} = (\Theta, h)$ fondée sur la construction de Merkle-Damgård, et soit $\mathcal{F} = (\Theta, f)$ la famille de fonctions de compression utilisée. Si la famille de fonctions de compression $\mathcal{F}$ est résistante aux collisions, alors la famille $\mathcal{H}$ est résistante aux collisions.*

*Preuve.* Nous commençons par prouver que pour tout $k$, une collisions pour $h_k$ conduit à une collision pour $f_k$. Soit $x$ et $x'$ deux chaînes de bits de longueur $b$ et $b'$ telles que $h_k(x) = h_k(x')$. Nous devons considérer deux cas $b \neq b'$ et $b = b'$.

- Cas $b \neq b'$ :
  Dans ce cas, la dernière étape du calcul de $h_k(x)$ est $H_{t+1} = f_k(H_t \parallel x_{t+1})$ et la dernière étape du calcul de $h_k(x')$ est $H'_{t+1} = f_k(H'_{t'} \parallel x'_{t'+1})$. Et comme $h_k(x) = h_k(x')$ on a :

  $$f_k(H_t \parallel x_{t+1}) = f_k(H'_{t'} \parallel x'_{t'+1}).$$

  Cependant, $b \neq b'$ d'où $x_{t+1} \neq x'_{t'+1}$ et donc :
  $H_t \parallel x_{t+1}$ et $H'_{t'} \parallel x'_{t'+1}$ sont deux chaînes de bits différentes qui forment une collision pour $f_k$.
- Cas $b = b'$ :
  Notons tout d'abord que si $b = b'$ alors $x_{t+1} = x'_{t+1}$. Comme $x \neq x'$ et $b = b'$, il existe au moins un indice $i$ pour lequel $x_i \neq x'_i$. Soit $i^* \leq t+1$ l'indice le plus élevé pour lequel $H_{i^*-1} \parallel x_{i^*} \neq H'_{i^*-1} \parallel x'_{i^*}$.
  Si $i^* = t+1$ alors $H_t \parallel x_{t+1} \neq H'_t \parallel x'_{t+1}$ sont deux chaînes de bits différentes qui forment une collision pour $f_k$ car :

  $$f_k(H_t \parallel x_{t+1}) = H_{t+1} = h_k(x) = h_k(x') = H'_{t+1} = f_k(H'_t \parallel x_{t+1}).$$

  Si $i^* \leq t$ alors le fait que $i^*$ soit l'indice le plus élevé implique que $H_{i^*} = H'_{i^*}$. En conséquence, $H_{i^*-1} \parallel x_{i^*} \neq H'_{i^*-1} \parallel x'_{i^*}$ sont deux chaînes de bits différentes qui forment une collision pour $f_k$.

Donc étant donné une collision pour $h_k$, nous pouvons extraire une collision pour $f_k$, clairement en temps polynomial. Nous étendons à présent ce résultat pour donner une réduction de sécurité.

Soit $\mathcal{A}$ un adversaire pour l'expérience de recherche de collision $Coll_{\mathcal{A},\mathcal{H}}(n)$. Nous construisons un adversaire $\mathcal{A}'$ pour l'expérience $Coll_{\mathcal{A},\mathcal{F}}(n)$ à partir de la définition suivante : Lorsque le paramètre $k$ est donné à l'adversaire $\mathcal{A}'$, celui-ci est transmis à l'adversaire $\mathcal{A}$. Nous récupérons le couple $(x, x')$ fournit par $\mathcal{A}$. Si $x \neq x'$ et $h_k(x) = h_k(x')$ alors nous extrayons le couple $(y, y')$ qui forme une collision pour $f_k$ de la façon décrite précédemment. Dans le cas contraire, l'adversaire échoue à produire une collision. Il est évident compte tenu de la définition de l'expérience de recherche de collision que l'adversaire $\mathcal{A}'$ réussi à produire une collision exactement lorsque l'adversaire $\mathcal{A}$ réussi. On a donc :

$$Pr[Coll_{\mathcal{A}',\mathcal{F}}(n) = 1] = Pr[Coll_{\mathcal{A},\mathcal{H}}(n) = 1] \leq g(n).$$

Il en découle que si la famille $\mathcal{F}$ est résistante aux collisions, alors la famille $\mathcal{H}$ est résistante aux collisions. □

Il est important de noter que la réciproque de ce théorème est fausse. Il est tout à fait possible d'envisager l'existence de fonctions de hachage résistantes aux collisions construites à partir de fonctions de compression qui ne soient pas résistantes aux collisions et dont le domaine est étendu avec l'algorithme de Merkle-Damgård.

**Attaque par extension (*Extension Attack*).** L'attaque par extension n'est pas à proprement parler une attaque ; il s'agit en fait d'une faiblesse inhérente à la structure itérative de l'algorithme de Merkle-Damgård. Soit une fonction de hachage $h$, utilisant l'algorithme de Merkle-Damgård comme extenseur de domaine. La connaissance de l'empreinte $h(M)$ d'un message $M$ qui après application du rembourrage donne $X = X_1, \ldots, X_t$, peut être utilisée pour construire l'empreinte d'un message particulier $M' = X||N$, où $N$ est quelconque. En effet, cette empreinte $h(M)$ constitue la sortie de la dernière itération de la fonction de compression $f$ : $h(M) = f(H_{t-1}, X_t)$. On peut donc calculer la valeur de $h(M')$ en prenant $h(M)$ comme nouvelle valeur initiale.

Cette propriété invalide l'utilisation de l'algorithme de Merkle-Damgård pour la construction d'un MAC-préfixe [PVO96]. De plus, cette propriété permet de distinguer trivialement la fonction $h$ d'un oracle aléatoire.

L'attaque par extension ne constitue pas la seule vulnérabilité de l'algorithme de Merkle-Damgård. La section 2-5.2 présente un certains nombres de vulnérabilités qui s'appliquent à cet algorithme.

### 2-3.2 Autres extenseurs de domaine

Pour palier aux vulnérabilités de l'algorithme de Merkle-Damgård, de nouvelles propositions d'extenseurs de domaine ont vu le jour ces dernières années. Les algorithmes Wide-Pipe et Double-Pipe [Luc05] ont été introduits par Lucks en 2005 pour résister aux attaques de multi-collisions. Ces constructions consistent essentiellement à augmenter la taille de la variable de chaînage. Le système HAIFA [BD06] de Biham et Dunkelman consiste à ajouter un sel (*Salt*) et un compteur à chaque itération de la fonction de compression. Ces propositions ont pour objectif de rendre les constructions itératives résistantes aux nouvelles attaques génériques apparues depuis la proposition de l'algorithme de Merkle-Damgård.

Une autre direction de recherche, plus théorique, consiste à proposer des extenseurs de domaine susceptibles de préserver le plus grand nombre possible de propriétés de la fonction de compression. Les propositions EMD [BR06] et ROX [ANP07] en sont des exemples, pour lesquels les auteurs ont démontré les qualités de leurs algorithmes. Cependant afin de rédiger des preuves formelles, ces propositions se fondent non pas sur une unique fonction de compression mais sur une famille de fonctions de compressions.

**Éponges cryptographiques (*Cryptographic Sponges*).** Les constructions de type éponges cryptographiques ont été introduites en 2006 par Bertoni *et al.* [BDP06]. L'objectif affiché de ces constructions est de fournir une fonction de hachage aussi proche que possible d'un oracle aléatoire. Il s'agit d'une construction itérative, mais

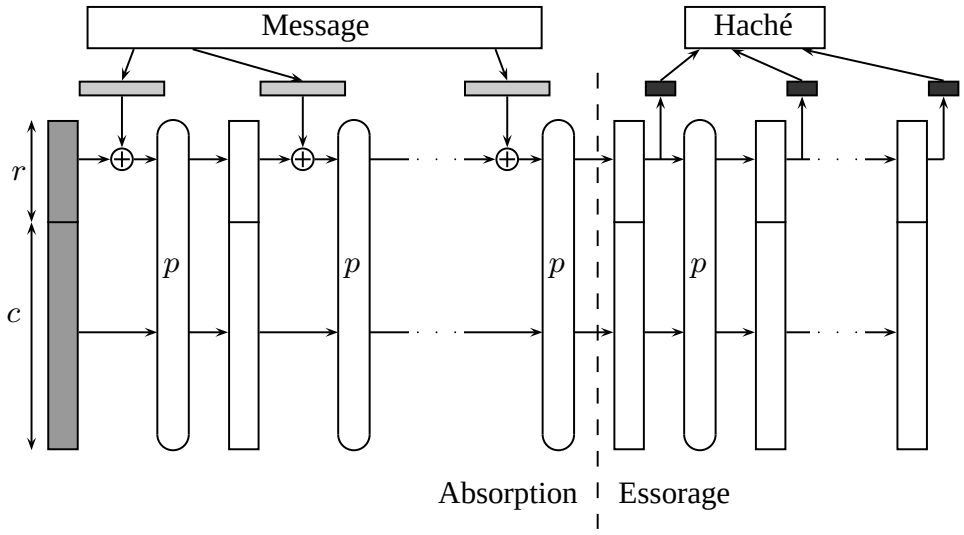



FIG. 2-4 – Éponges cryptographiques.

fondée sur une transformation de longueur fixe $p$ (plus précisément une permutation) à la place d'une fonction de compression. Une fonction éponge prend comme argument en entrée une chaîne de bits de longueur variable et produit en sortie une chaîne de bits de longueur arbitraire.

Cette construction opère sur un état constitué de $b = r + c$ bits, où le nombre $r$ est appelé taux de bits (*Bitrate*) et le nombre c la capacité (*Capacity*). Le message à traiter est tout d'abord rembourré et découpé en blocs de $r$ bits. La construction éponge procède alors en deux étapes :

- une première étape d'absorption qui consiste à "remplir" l'éponge en utilisant les blocs de message,
- puis une seconde étape d'essorage (*Squeezing*) qui consiste à "presser" l'éponge pour obtenir en sortie le nombre de bits désiré.

La capacité $c$ détermine le niveau de sécurité susceptible d'être atteint par la fonction de hachage. Une illustration graphique du principe des éponges cryptographiques se trouve figure 2-4.

Les arguments de sécurité de ce type de construction sont fondés sur le modèle de l'indifférentiabilité (*Indifferentiability Framework*) qui a été introduit par Maurer *et al.* [MRH04] et que nous décrivons section 2-5.3. Les auteurs ont montré dans [BDP08a] que si la fonction $p$ est modélisée en tant que permutation aléatoire ou en tant qu'oracle aléatoire, la construction éponge est indifférentiable d'un oracle aléatoire monolithique.

## 2-4 Autres constructions

Les constructions itératives bien qu'employées dans une très large majorité des fonctions de hachage cryptographiques, ne sont pas les seules constructions à avoir été proposées. On peut citer par exemple la construction du hachage par paquet (*Bucket Hashing*) introduite en 1995 par Rogaway [Rog95]. Le paradigme de randomi-

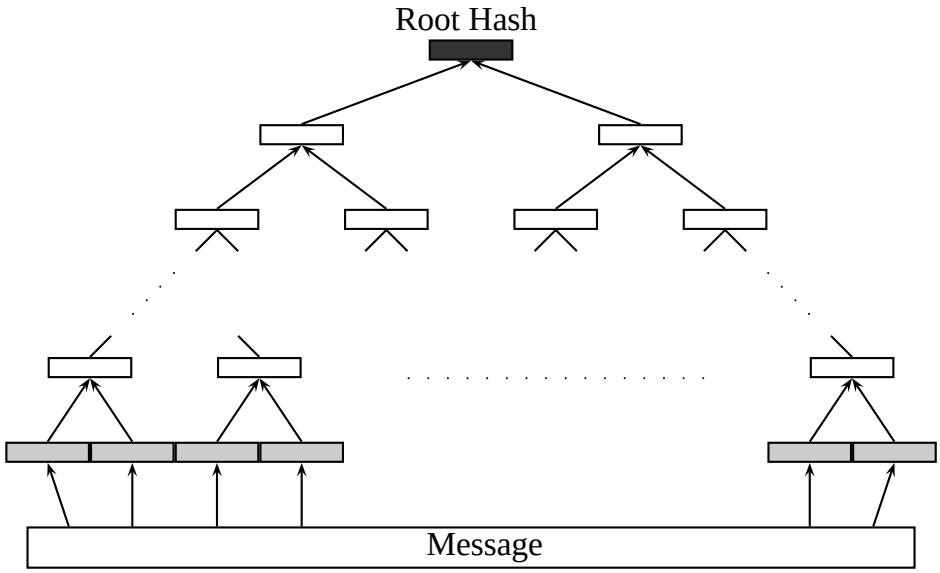



FIG. 2-5 – Arbres de Merkle.

sation puis combinaison (*Randomize-then-Combine Paradigm*) introduit par Bellare et Micciancio [BM97] constitue un autre exemple. La fonction XOR-Hash, décrite au chapitre 9, est fondée sur ce paradigme.

**Arbres de Merkle.** Les arbres de Merkle ont été introduits par Ralph Merkle en 1979 [Mer79b], dans le but de construire un schéma de signature électronique fondé sur les fonctions de hachage. Mais des primitives cryptographiques plus efficaces, fondées sur la théorie des nombres (RSA, DSA, ECC), ont rendu obsolète les arbres de Merkle pour ce domaine d'utilisation. Le principe des arbres de Merkle consiste à construire un arbre de hachage à partir d'une fonction de hachage et de données. Les feuilles de l'arbre sont les empreintes des différents blocs de données ; les noeuds de l'arbre sont constitués par les empreintes de leur fils respectifs. Au sommet de l'arbre, on trouve l'empreinte racine (*Root Hash*). Une illustration graphique du principe des arbres de Merkle se trouve figure 2-5. La plupart des implémentations d'arbre de Merkle utilisent des arbres binaires, mais le principe peut être étendu aux arbres $q$-aires.

Les arbres de Merkle sont intensivement employés dans les réseaux pair à pair (*Peer-to-Peer Networks*). Avant de commencer le téléchargement d'un fichier, l'empreinte racine est obtenue d'une source de confiance : par exemple, le site internet fournisseur de l'application pair à pair. Lorsque l'empreinte racine a été obtenue, l'arbre de hachage peut être reçu et vérifié à partir de n'importe laquelle des sources du réseau. Cet arbre permet de vérifier l'intégrité des données au fur et à mesure de la transmission des différentes parties d'un fichier. Il suffit de connaître le noeud se situant à la hauteur correspondante dans l'arbre, pour vérifier une sous-branche de l'arbre ou un fragment de fichier. Un autre avantage des arbres de Merkle réside dans leur nature parallélisable.

## 2-5 Sécurité des constructions

### 2-5.1 Attaques génériques

Nous détaillons ici les deux types d'attaques génériques susceptibles d'être appliquées aux fonctions de hachage. Ces attaques s'appliquent à toutes les fonctions de hachage indépendamment de la façon dont elles sont construites. Elles constituent les références à laquelle sont comparées les complexités des attaques visant à mettre en défaut une des trois propriétés classiques.

**Paradoxe des anniversaires.** L'attaque du paradoxe des anniversaires est une attaque générique qui permet d'obtenir une paire de messages formant une collision. Soit $h$ une fonction de hachage de $\{0,1\}^*$ vers $\{0,1\}^n$. L'adversaire construit deux ensembles $E_1$ et $E_2$, de taille respectivement $\#E_1$ et $\#E_2$, constitués de couples de messages tous différents et de leurs empreintes. La probabilité qu'un message appartenant à $E_1$ et un message appartenant à $E_2$ possèdent la même empreinte est

$$P \approx 1 - \exp \frac{\#E_1 \times \#E_2}{2^n}.$$

En particulier, si $\#E_1 = \#E_2 = 2^{n/2}$ alors

$$P \approx 1 - \frac{1}{e} = 0,63.$$

La complexité algorithmique d'une telle attaque est donc de l'ordre de $\mathcal{O}(2^{n/2})$ évaluations de la fonction de hachage.

Une fonction de hachage cryptographique est considérée comme "cassée" dès lors que l'on peut produire une attaque par collision possédant une complexité inférieure à l'attaque du paradoxe des anniversaires.

**Attaques par force brute.** Ce type d'attaque concerne les propriétés de résistance à la recherche d'antécédent et de second antécédent. Soit $h$ une fonction de hachage de $\{0,1\}^*$ vers $\{0,1\}^n$. L'adversaire tire aléatoirement un message, calcule son empreinte et vérifie si celle-ci correspond à l'empreinte désirée. Si la fonction de hachage se comporte comme une fonction pseudo-aléatoire, la probabilité de succès est égale à $1/2^n$. La probabilité de succès d'une telle attaque peut être améliorée en augmentant le nombre de messages tirés. Si l'adversaire calcule l'empreinte de $2^n$ messages différents, la probabilité de succès est approximativement égale à $0,63$.

En pratique, ce type d'attaque est parallélisable. De plus, un adversaire peut construire cette attaque simultanément pour un sous-ensemble d'empreintes possibles, avec pour objectif de ne trouver qu'un seul antécédent. La complexité de l'attaque est alors réduite par la taille du sous-ensemble visé.

De la même façon que pour l'attaque par paradoxe des anniversaires, une fonction de hachage est considérée comme cassée dès lors qu'il existe une attaque par recherche d'antécédent, ou de second antécédent, possédant une complexité meilleure que $\mathcal{O}(2^n)$ évaluations de la fonction de hachage.

### 2-5.2 Attaques spécifiques

Les attaques génériques décrites dans la section précédente s'appliquent à toutes les fonctions de hachage. Lors de la description de l'algorithme de Merkle-Damgård nous avons remarqué que cet algorithme, bien que le plus populaire, possède certaines vulnérabilités. Nous décrivons ici, les attaques développées spécifiquement contre l'algorithme de Merkle-Damgård. Ces attaques sont susceptibles d'être appliquées à d'autres fonctions de hachage itératives.

**Attaque des Multi-collisions.** Dans une tentative d'augmenter la sécurité, il a été proposé de créer une nouvelle fonction de hachage en concaténant les sorties de deux fonctions de hachage différentes. Soient deux fonctions de hachage résistantes aux collisions : $h$ produisant des empreintes de taille $n$ bits et $g$ produisant des empreintes de taille $m$ bits. La fonction de hachage $hg$ définie par $hg(M) = h(M)||g(M)$ produit des empreintes de taille $n+m$ bits. La fonction $hg$ est résistante aux collisions, si on ne peut construire une attaque possédant une complexité inférieure à $\mathcal{O}(2^{\frac{n+m}{2}})$ évaluations de la fonction de hachage $hg$.

Cependant, Antoine Joux a démontré en 2004 [Jou04] que si l'une au moins des deux fonctions concaténées est une fonction itérative, il existe une attaque strictement meilleure que l'attaque générique. Cette attaque repose sur le principe des multi-collisions. Une $k$-multi-collision est un ensemble de $k$ messages possédant la même empreinte (le cas où $k = 2$ correspond à une collision classique). Sans perte de généralité on peut supposer que la fonction $h$ est la fonction itérative fondée sur l'algorithme de Merkle-Damgård. L'attaque des multi-collisions consiste à construire une $m/2$-multi-collision pour $h$. La première étape consiste à obtenir une collision pour la fonction de compression de $h$ avec comme variable de chaînage l'$IV$, cette collision est obtenue après $\mathcal{O}(2^{n/2})$ évaluations de la fonction de compression de $h$. A partir de la nouvelle variable de chaînage on construit, de la même façon, une deuxième collision pour la fonction de compression ; on répète l'opération $m/2$ fois. Ces $m/2$ collisions aboutissent à $2^{m/2}$ messages différents possédant la même empreinte, soit une $m/2$-multi-collision pour $h$ pour un coût égal à $\mathcal{O}(\frac{m}{2} \times 2^{n/2})$ évaluations de la fonction de compression de $h$. De plus, parmi ces $2^{m/2}$ messages différents, le paradoxe des anniversaires nous indique qu'il y a une forte probabilité que deux d'entre eux constituent une collision pour $g$. Finalement, on obtient une collision pour la fonction $hg$ avec un coût inférieur à $\mathcal{O}(2^{\frac{n+m}{2}})$ évaluations de cette fonction. Une illustration de l'attaque des multi-collisions est proposée figure 2-6.

**Amélioration de la recherche de second antécédent.** Pour une fonction de hachage idéale produisant des empreintes de taille $n$ bits, une attaque par recherche de second antécédent nécessite $\mathcal{O}(2^n)$ évaluations de la fonction de hachage. Cependant, Dean [Dea99] a remarqué que cette propriété n'est pas satisfaite par les fonctions de hachage itératives dès lors que l'on peut facilement trouver des points fixes pour la fonction de compression. Kelsey et Schneier [KS05] ont montré que l'on peut construire un second antécédent en moins de $\mathcal{O}(2^n)$ évaluations de la fonction de compression sans utiliser de points fixes. Pour des messages longs constitués de $2^k$ blocs, il est possible de trouver un second antécédent avec une complexité de l'ordre de $\mathcal{O}(k \times 2^{n/2} + 1 + 2^{n-k+1})$ évaluations de la fonction de compression.

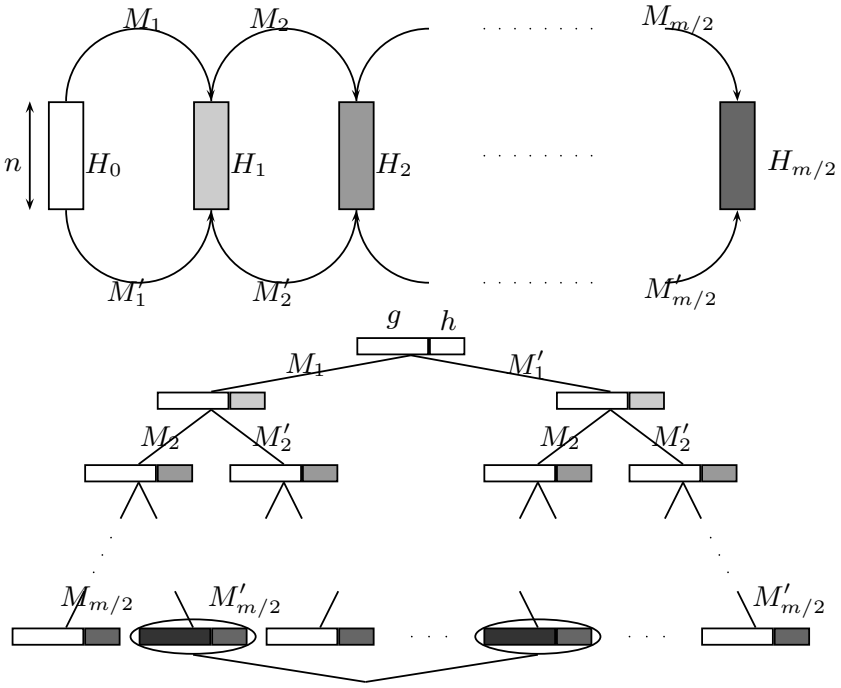



FIG. 2-6 – Principe de l'attaque des multi-collisions.

Le principe de cette attaque s'appuie sur des "messages extensibles" (*Expandable Messages*) qui sont des ensembles de messages de longueurs différentes. Ils sont utilisés pour construire des messages de taille variable conduisant à une même variable de chaînage avant insertion du bloc contenant le codage de la longueur. On peut considérer un message extensible comme une multi-collision où les messages possèdent un nombre de blocs différent, mais conduisent tous à une même variable de chaînage.

Un message extensible produisant des messages de taille variant entre $k$ et $2^k + k + 1$ blocs permet de trouver un second antécédent pour un message long composé de $2^k + k + 1$ blocs. L'attaque consiste à stocker l'ensemble des variables de chaînage intermédiaires puis à tenter de lier la variable de chaînage finale du message extensible à l'une des variables de chaînage du message original. Lorsqu'un lien est trouvé, il est toujours possible d'ajuster le message extensible afin qu'il présente la même longueur que la première partie du message original, de son début jusqu'à l'endroit où le lien a été trouvé. On obtient donc deux messages longs de même longueur et présentant une collision des variables de chaînage à un certain point. On concatène alors au message extensible la fin du message original ce qui conduit à l'obtention de la même empreinte. Ce processus est illustré figure 2-7.

La complexité totale est égale à $\mathcal{O}(k \times 2^{n/2} + 1 + 2^{n-k+1})$. Le premier terme est dû à la génération du message extensible, le second décrit la complexité nécessaire pour trouver un lien entre le message extensible et une des variables de chaînage du message original.

**Attaque de Nostradamus (*Herding Attack*).** L'attaque de Nostradamus introduite par Kelsey et Kohno[KK06] permet à un adversaire de s'engager, après avoir

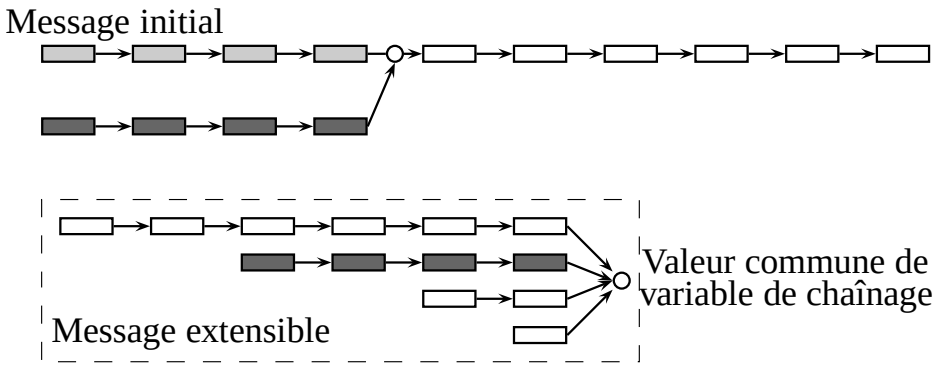



FIG. 2-7 – Attaque de Kelsey et Shneier.

accompli une étape de pré-calcul, sur une certaine valeur constituée par l'empreinte d'un message publié à posteriori. Plus tard, l'adversaire produit pour tout message, un suffixe concordant avec cet engagement. Le principe de cette attaque repose sur une structure de données pré-calculée qui permet à $2^k$ séquences de blocs de message de converger de façon itérative vers la même empreinte. Pour une fonction de hachage fondée sur l'algorithme de Merkle-Damgård, le coût de cette attaque est de $\mathcal{O}(2^{\frac{n+k}{2}+2})$ (en pré-calcul) et de $\mathcal{O}(2^{n-k})$ (en ligne) évaluations de la fonction de compression.

Lors de l'attaque, l'adversaire s'engage sur une empreinte publique $e$ obtenue à partir d'un fonction de hachage itérative $h$ fondée sur une fonction de compression $f$. Après la phase d'engagement, un challenge est proposé sous la forme d'un préfixe $P$ sur lequel l'adversaire ne possède aucun contrôle. Il doit alors produire un suffixe $S$ tel que $h(P||S) = e$. La valeur $e$ est choisie spécifiquement par l'adversaire après une phase de pré-calcul. Ce pré-calcul consiste à enregistrer des variables de chaînage à partir desquelles l'adversaire sait pouvoir atteindre la valeur $e$. La structure de données utilisée pour enregistrer les variables de chaînage est un arbre binaire construit à l'aide d'un algorithme de recherche de collision. En utilisant par exemple le paradoxe des anniversaires, on peut créer une structure en diamant avec $2^{k+1} - 2$ variables de chaînage intermédiaires, pour un coût de l'ordre de $\mathcal{O}(2^{\frac{n+k}{2}+2})$ évaluations de la fonction de compression $f$. Une illustration de la structure en diamant est donnée figure 2-8.

Dans la phase en ligne de l'attaque, l'adversaire recherche de façon exhaustive un suffixe $S'$ tel que $h(P||S')$ forme une collision avec une des variables de chaînage intermédiaires. Cette étape requiert de tester de l'ordre de $\mathcal{O}(2^{n-k})$ valeurs pour $S'$. Une fois cette collision trouvée, l'adversaire peut obtenir à partir de la structure en diamant un message $Q$, et construire le suffixe $S = S'||Q$, de taille $k + 1$ blocs, tel que $h(P||S) = e$.

Andreeva *et al.*[ABD09, ABF08] ont proposé plusieurs extensions et généralisations pour cette attaque.

### 2-5.3 Modèle de l'indifférentiabilité

Le modèle de l'oracle aléatoire [BR93] est traditionnellement employé afin de construire des preuves de sécurité pour les protocoles cryptographiques. Dans ce

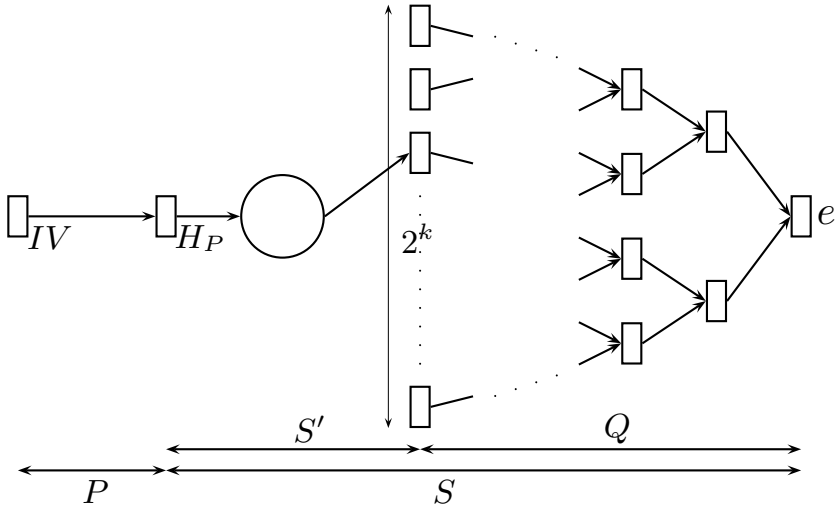



FIG. 2-8 – Attaque de Kelsey et Kohno.

modèle, on donne à toutes les parties impliquées dans le protocole, y compris l'adversaire, un accès à une "vraie" fonction aléatoire. Mais une telle fonction n'existe pas en pratique. Aussi, lors de la mise en oeuvre du protocole, cette fonction aléatoire est instanciée sous la forme d'une fonction de hachage idéale. Les preuves formelles obtenues dans le modèle de l'oracle aléatoire indiquent qu'il n'existe pas de défaut structurel dans le design du protocole. Cependant, de récents résultats [CGH98, Nil02] montrent qu'il est théoriquement possible de construire des cas pathologiques qui sont sûrs dans ce modèle mais complètement non-sûrs dans le modèle standard. Malgré cela, le modèle de l'oracle aléatoire reste un outil essentiel pour le design de protocoles.

Des chercheurs se sont intéressés à l'application du modèle de l'oracle aléatoire au design de fonctions de hachage, et plus particulièrement aux fonctions de hachage itératives. C'est le cas de Coron *et al.* [CDM05] qui se sont penchés sur l'algorithme de Merkle-Damgård. Pour ce faire, ils ont utilisé le modèle de l'indifférentiabilité (*Indifferentiability Framework*) introduit par Maurer *et al.* [MRH04] en 2004.

De par sa nature, une fonction de hachage itérative est trivialement distinguable d'un oracle aléatoire, en utilisant par exemple une attaque par extension (décrite dans la chapitre suivant). Ceci est dû au fait qu'un oracle aléatoire est une structure monolithique, constituée d'un seul bloc, contrairement à une fonction de hachage itérative. L'objectif du modèle de l'indifférentiabilité consiste donc à contourner ce problème à l'aide d'un simulateur. De façon informelle, pour prouver l'indifférentiabilité d'une construction $C$ fondée sur une primitive idéale $F$ par rapport à un oracle aléatoire $R$, on utilise un simulateur $S$. La fonction du simulateur $S$ consiste à simuler le comportement de la construction $C$, tout en maintenant une consistance avec l'oracle aléatoire $R$. Dès lors qu'il n'existe pas de distingueur $D$ capable de distinguer les sorties des couples $(C^F, F)$ et $(R, S^R)$, la construction $C$ est dites indifférentiable d'un oracle aléatoire. Une illustration graphique du principe du modèle de l'indifférentiabilité se trouve figure 2-9.

Ce modèle permet de montrer qu'un extenseur de domaine particulier n'introduit

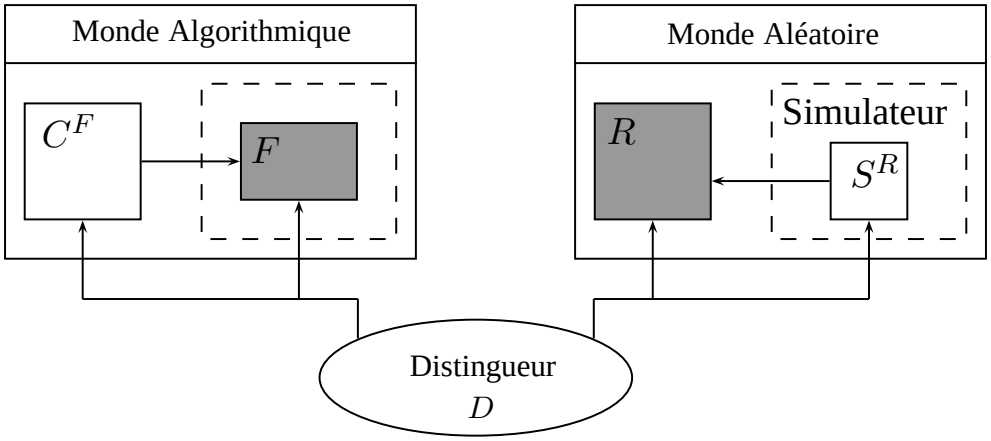



FIG. 2-9 – Modèle de l'indifférentiabilité. Les boîtes grisées représentent des primitives idéales, les boîtes vides des algorithmes.

pas de faiblesse structurelle supplémentaire, lorsqu'il est appliqué à une fonction de compression idéale. Bien qu'introduite récemment, l'indifférentiabilité est aujourd'hui une des propriétés formelles que se doivent de posséder les fonctions de hachage cryptographiques itératives.

## Deuxième partie

# Cryptanalyse des fonctions SHA-0 et SHA-1

# 3

# Présentation des fonctions SHA-0 et SHA-1

**Sommaire**



Les fonctions de hachage cryptographiques SHA-0 et SHA-1 ont toutes deux été crées par l'agence nationale de sécurité américaine (*National Security Agency*) et publiées par le NIST en tant que standard fédéral de traitement de l'information (*Federal Information Processing Standard*). Le premier standard baptisé SHA (*Secure Hash Standard*) fut adopté en 1993. Cependant en 1995, la NSA demanda une modification de la procédure d'expansion de message. Cette nouvelle version hérita de l'appellation SHA-1, et l'ancien standard fût rebaptisé SHA-0.

Dans ce chapitre, nous allons décrire le principe de fonctionnement commun aux deux fonctions de hachage SHA-0 et SHA-1, nous donnerons ensuite un bref historique de l'évolution de la sécurité de ces fonctions relativement aux attaques par collisions.

## 3-1 Principe de fonctionnement

### 3-1.1 Extenseur de domaine

Les fonctions SHA-0 et SHA-1 sont des fonctions de hachage itératives utilisant l'algorithme de Merkle-Damgård comme extenseur de domaine appliquée à une fonction de compression ad hoc.

Le message destiné à être haché est soumis à un rembourrage (*Padding*) et découpé en $k$ blocs $(M_0, \ldots, M_{k-1})$ de taille 512 bits. À chaque itération de la fonction de compression $h$, une variable de chaînage $H_t$ de taille 160 bits est mise à jour à partir d'un bloc de message $M_{t+1}$, *i.e* $H_{t+1} = h(H_t, M_{t+1})$. La variable de chaînage initiale $H_0$ (*Initial Value*) est fixe et prédéfinie ; la sortie de la fonction de hachage est la dernière variable de chaînage $H_k$.

| A | B | C | D | E |
|---|---|---|---|---|
| 0x67452301 | 0xefcdab89 | 0x98badcfe | 0x10325476 | 0xc3d2e1f0 |

TAB. 3-1 – Valeurs d'initialisation utilisées dans SHA-0 et SHA-1.

**Rembourrage (*Padding*).**

Le rembourrage utilisé pour les fonctions SHA-0 et SHA-1 consiste à ajouter un 1 après le dernier bit du message puis à ajouter autant de 0 que nécessaire afin d'obtenir un nombre entier de blocs de 512 bits, avec la condition supplémentaire que les 64 derniers bits du dernier bloc contiennent la représentation binaire de la taille du message exprimée en nombre de bits. Cette taille est donc limitée à un maximum de $2^{64} - 1$ bits. En plus de fournir un nombre entier de blocs de 512 bits à la fonction de compression, ce rembourrage permet de garantir qu'aucun message ne puisse constituer le préfixe d'un autre message.

**Valeurs d'initialisation (*Initial Values*).**

Les valeurs prédéfinies pour la variable de chaînage initiale sont identiques pour SHA-0 et SHA-1. Elles sont définies Table 3-1.

### 3-1.2 Fonction de compression

La fonction de compression utilisée est construite sur le principe de Davis-Meyer. Elle utilise une fonction ad-hoc $\mathcal{E}$ comme schéma de chiffrement par bloc, où les variables de chaînage successives constituent les paires clair/chiffré et où les différents blocs de message jouent le rôle de clé. Un rebouclage (*Feed-Forward*) est utilisé afin d'éviter une inversion directe de la fonction de compression. On a donc :

$$H_{t+1} = \mathcal{E}(H_t, M_{t+1}) \boxplus H_t,$$

où la notation $\boxplus$ dénote une addition modulo $2^{32}$ mot de 32 bits par mot de 32 bits.

La fonction $\mathcal{E}$ se fonde sur un état interne mis à jour selon un schéma de Feistel asymétrique généralisé. Une représentation graphique de cette fonction se trouve figure 3-1.

Elle est composée de 80 pas (4 rondes de 20 pas), chaque pas utilise un mot de 32 bits $W_i$ (soit un total de $80 \times 32 = 2560$ bits) pour mettre à jour 5 registres internes $(A, B, C, D, E)$ de 32 bits chacun. Étant donné que plus de bits de message que disponibles sont nécessaires, une procédure d'expansion récursive est utilisée pour générer les bits surnuméraires. La différence entre les fonctions SHA-0 et SHA-1 réside dans cette procédure d'expansion.

**Expansion de message.**

Tout d'abord, le bloc de message courant $M_t$ est découpé en 16 mots de 32 bits numérotés de 0 à 15 : $W_0, \ldots, W_{15}$. Ces 16 mots sont ensuite étendus selon les formules suivantes :

$$SHA-0 : \quad W_i = W_{i-16} \oplus W_{i-14} \oplus W_{i-8} \oplus W_{i-3} \text{ avec } 16 \leq i \leq 79. \qquad (3\text{-}1)$$

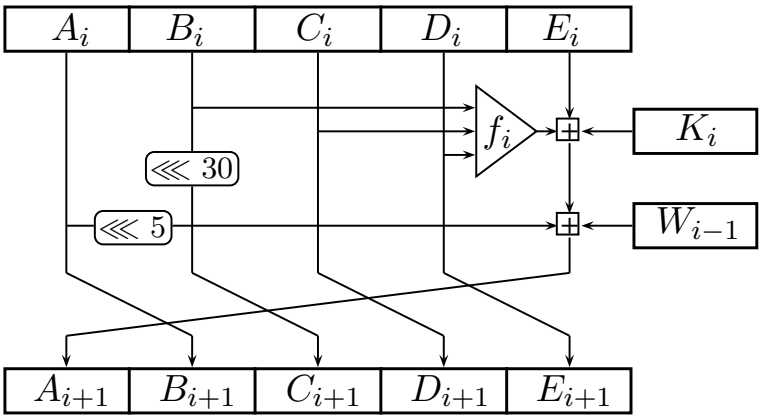



FIG. 3-1 – Fonction de mise à jour des registres commune à SHA-0 et SHA-1.

| ronde | pas $i$ | $f_i(B,C,D)$ | $K_i$ |
|---|---|---|---|
| 1 | $1 \leq i \leq 20$ | $f_{IF} = (B \wedge C) \oplus (\overline{B} \wedge D)$ | `0x5a827999` |
| 2 | $21 \leq i \leq 40$ | $f_{XOR} = B \oplus C \oplus D$ | `0x6ed6eba1` |
| 3 | $41 \leq i \leq 60$ | $f_{MAJ} = (B \wedge C) \oplus (B \wedge D) \oplus (C \wedge D)$ | `0x8fabbcdc` |
| 4 | $61 \leq i \leq 80$ | $f_{XOR} = B \oplus C \oplus D$ | `0xca62c1d6` |

TAB. 3-2 – Fonctions booléènes et constantes utilisées dans SHA-0 et SHA-1.

$$SHA-1 : \quad W_i = (W_{i-16} \oplus W_{i-14} \oplus W_{i-8} \oplus W_{i-3}) \lll 1 \text{ avec } 16 \leq i \leq 79. \quad (3\text{-}2)$$

où la notation $\lll 1$ signifie une permutation circulaire de 1 bit vers la gauche. Nous verrons dans la partie consacrée aux cryptanalyses de SHA-0 et SHA-1 que cette modification, d'apparence mineure, a un impact majeur sur la sécurité et déterminant sur ces cryptanalyses.

**Mise à jour des registres.**

En premier lieu, la variable de chaînage $H_t$ est découpée en 5 mots de 32 bits afin de fournir les états initiaux $(A_0, B_0, C_0, D_0, E_0)$ des 5 registres. Alors, la transformation suivante est appliquée 80 fois, $i$ variant de 0 à 79 :

$$PAS_{i+1} : \begin{cases} A_{i+1} = (A_i \lll 5) + f_i(B_i, C_i, D_i) + E_i + K_i + W_i, \\ B_{i+1} = A_i, \\ C_{i+1} = B_i \ggg 2, \\ D_{i+1} = C_i, \\ E_{i+1} = D_i. \end{cases}$$

où les $K_i$ sont des constantes de ronde prédéterminées et les $f_i$ des fonctions booléènes définies table 3-2.

**Rebouclage (*Feed-Forward*).**

Les sommes modulo $2^{32}$ : $(A_0+A_{80})$, $(B_0+B_{80})$, $(C_0+C_{80})$, $(D_0+D_{80})$, $(E_0+E_{80})$ sont concaténées pour former la variable de chaînage suivante $H_{t+1}$.

**Remarque.**

On remarque que pour un pas $i$, les états de tous les registres ne sont que des copies circulairement permutées de différents états du registre $A$. On peut donc considérer uniquement l'évolution des états du seul registre $A$ pour chaque pas. En conséquence, il est possible de récrire la fonction de mise à jour des registres sous la forme suivante :

$$A_{i+1} = (A_i \lll 5) + f_i(A_{i-1}, A_{i-2} \ggg 2, A_{i-3} \ggg 2) + A_{i-4} \ggg 2 + K_i + W_i.$$

Nous serons amenés dans la suite, à favoriser l'une ou l'autre forme d'écriture de la fonction de mise à jour des registres dans l'objectif de choisir la plus claire façon de présenter l'information.

## 3-2 Sécurité de la fonction SHA-0

Publiée en 1993, la fonction SHA-0 fut mise à jour par la *National Security Agency* (NSA) en 1995 pour donner la fonction SHA-1. Le but de la modification apportée à la fonction d'expansion de message ne fut jamais publié, bien qu'il soit facile de voir que l'introduction de la permutation circulaire assure une meilleure diffusion. C'est en 1998 que Chabaud et Joux [CJ98] publièrent la première attaque par collision théorique contre SHA-0, avec une complexité évaluée à $2^{61}$ appels à la fonction de compression. Il s'agit d'une cryptanalyse différentielle fondée sur une propriété particulière de la fonction de mise à jour des registres dénommée collision locale. Pour la fonction SHA-1, l'introduction d'une meilleure diffusion au niveau de l'expansion du message, permettait d'éviter une transposition directe de l'attaque de Chabaud et Joux.

Le principe de cette attaque fût repris en 2004 par Biham et Chen [BC04] qui introduisirent une technique d'accélération de recherche fondée sur l'existence de bits neutres (décrits section 5-6). L'utilisation des bis neutres permit de produire des quasi-collisions pour SHA-0. L'année suivante la première collision fut obtenue par une équipe de chercheurs [BCJ05] après cinq semaines de calculs sur de nombreux ordinateurs. La complexité de cette attaque est évaluée à $2^{51}$ évaluations de la fonction de compression de SHA-0. La technique utilisée s'appuie sur les deux cryptanalyses précédentes et l'utilisation de la technique des blocs multiples présentée section 5-5.

En 2005, les travaux de Wang *et al.* [WYY05d] introduisirent une nouvelle approche de la cryptanalyse différentielle de SHA-0 ainsi que des techniques supplémentaires (caractéristique non-linéaire détaillée section 5-4, modifications de message décrites section 5-6) qui permirent d'obtenir une collision avec une complexité annoncée à $2^{39}$ appels de la fonction de compression. Naito *et al.* [NSS06] proposèrent en 2006 de nouvelles modifications de message permettant une amélioration théorique de l'attaque.

La meilleure attaque par collision publiée contre la fonction SHA-0 est l'attaque que nous avons menée en commun avec Thomas Peyrin [MP08] qui est décrite au

chapitre 6 de cette thèse. La complexité en ligne de cette attaque correspond à environ $2^{33}$ appels à la fonction de compression de SHA-0.

Le standard SHA-0, trop rapidement modifié en SHA-1, n'a pratiquement pas été utilisé par l'industrie.

## 3-3 Sécurité de la fonction SHA-1

La fonction SHA-1 a longtemps résisté aux cryptanalystes ce qui lui a valu de devenir la fonction de hachage cryptographique la plus utilisée dès lors qu'un impératif de sécurité était nécessaire. En effet, les attaques contre la fonction SHA-0 menées par Chabaud et Joux, et Biham et Chen n'étaient applicables qu'à des versions très réduites de SHA-1. Cependant, ce ne fût pas le cas des travaux de Wang *et al.* [WYY05b]. Les nouvelles techniques introduites dans leur article ont été transposées efficacement et ont permis de proposer la première attaque par collision théorique avec une complexité revendiquée égale à $2^{69}$ évaluations de la fonction de compression. Une nouvelle complexité pour l'attaque fût ensuite annoncée à $2^{63}$ appels de la fonction de compression [WYY05a, WYY05c].

S'étayant sur ces travaux, de nouvelles cryptanalyses ont été menées. De Cannière et Rechberger [DCR06] ont exhibé une première collision pour une version réduite à 64 pas, bientôt suivie de collisions pour une version réduite à 70 pas, par De Cannière *et al.* [DCM07] et Joux et Peyrin [JP07]. Un calcul distribué a été lancé à l'initiative de Mendel *et al.* [MRR07], avec une complexité annoncée inférieure à $2^{61}$ appels de la fonction de compression de SHA-1 ; ce calcul n'a toujours pas abouti.

Au moment ou nous rédigeons cette thèse, deux nouvelles collisions ont été soumises sur le serveur ePrint [Gre10]. La première s'applique à une version réduite à 72 pas avec une complexité de $2^{50}$ appels de la fonction de compression et la seconde à une version réduite à 73 pas avec une complexité de $2^{52}$ appels de la fonction de compression. Aucune collision pour la version normalisée de SHA-1 n'a encore été publiée.

L'utilisation de la fonction SHA-1 en tant que standard de hachage pour le gouvernement américain a été interdite par le NIST après 2010, bien que son utilisation en tant que code d'authentification (HMAC), fonction de dérivation de clé et générateur pseudo-aléatoire soit encore autorisée.

# 4

# Principe des cryptanalyses de SHA-0 et SHA-1

**Sommaire**



## 4-1 Cryptanalyses différentielles des fonctions de hachage cryptographiques

En 1990, Eli Biham et Adi Shamir ont introduit la notion de cryptanalyse différentielle [BS90]. En utilisant cette nouvelle méthode, Biham et Shamir construisirent une attaque à clair choisi, strictement plus efficace que la recherche exhaustive, contre l'algorithme de chiffrement par blocs DES (*Data Encryption Standard*) [FIPS-DES]. La cryptanalyse différentielle traite spécifiquement des paires de textes chiffrés dont les textes clairs correspondants présentent certaines différences particulières. Elle analyse l'évolution de ces différences à travers les différentes rondes du DES. Le terme "différence" est défini, dans le cadre de la cryptanalyse du DES, par l'opération *ou exclusif* bit a bit.

La cryptanalyse différentielle appliquée à des algorithmes de chiffrement a pour but de retrouver la clé de chiffrement. Dans le cadre des fonctions de hachage,

on emploie la cryptanalyse différentielle pour obtenir une collision ou une quasi-collision. Les premières applications de la cryptanalyse différentielle aux fonctions de hachage cryptographiques furent celles menées contre N-HASH [MOI90] et SNEFRU [Mer90]. Ces deux attaques, menées par Biham et Shamir, sont décrites dans [BS91a] et [BS91b]. En 2004, Wang *et al.* [WFL04] ont appliqué avec succès ce type de cryptanalyse à plusieurs fonctions de hachage dont MD5 et RIPE-MD.

Le principe de l'attaque différentielle des fonctions de hachage consiste dans une première étape à déterminer un ensemble de différences (la différentielle) entre deux messages susceptibles de conduire à l'obtention d'une collision ou d'une quasi-collision. La seconde étape consiste à produire un couple de messages vérifiant cette différentielle et conduisant au résultat attendu. La complexité de ce type d'attaque est en général évaluée par l'inverse de la probabilité qu'un message tiré aléatoirement et qu'un message obtenu à partir du premier et de la différentielle, forment une collision (une quasi-collision). Cette complexité est, le plus souvent, exprimée en nombre d'instanciations de la fonction de hachage.

La cryptanalyse différentielle est aujourd'hui un outil prépondérant utilisé par les cryptologues pour construire des attaques contre les fonctions de hachage cryptographiques.

## 4-2 Modèle des collisions locales

Les attaques différentielles menées contre SHA-0 et SHA-1 ont pour point commun de se fonder sur une propriété particulière de la fonction de mise à jour des registres : les collisions locales. En effet, Wang [Wan97], ainsi que Chabaud et Joux [CJ98], ont montré de façon indépendante que l'introduction d'une perturbation à un pas $i$, pouvait être corrigée par l'introduction aux pas suivants d'autres perturbations appelées corrections.

### 4-2.1 Approximation linéaire de la fonction de mise à jour des registres

Pour illustrer cette propriété, nous allons nous intéresser à une version linéaire de la fonction de mise à jour des registres fondée sur deux hypothèses. La première hypothèse consiste à linéariser les fonctions de ronde $f_i$ et les additions modulo $2^{32}$ en remplaçant celles-ci par l'addition bit à bit modulo 2 ($xor$) sur des mots de 32 bits. Les formules de mise à jour des registres peuvent alors être énoncées de la façon suivante :

$$PAS_{i+1} : \begin{cases} A_{i+1} = A_i \lll 5 \oplus B_i \oplus C_i \oplus D_i \oplus E_i \oplus K_i \oplus W_i, \\ B_{i+1} = A_i, \\ C_{i+1} = B_i \lll 30, \\ D_{i+1} = C_i, \\ E_{i+1} = D_i, \end{cases}$$

soit :

$$A_{i+1} = A_i \lll 5 \oplus A_{i-1} \oplus A_{i-2} \lll 30 \oplus A_{i-3} \lll 30 \oplus A_{i-4} \lll 30 \oplus K_i \oplus W_i.$$

La seconde hypothèse considère que l'on peut intervenir sur les mots de message étendu $W_i$ indépendamment du processus d'expansion dont ils sont issus. Cela permet d'appliquer simplement, une perturbation à un bit $j$ d'un mot $W_{i-1}$ quel que soit le pas considéré $i$ de la fonction de compression.

L'idée fondamentale pour construire une collision locale consiste à perturber à un pas $i$ un mot $W_{i-1}$ et à examiner les effets de cette perturbation sur les états des registres des pas suivants. Un exemple de perturbation peut consister à effectuer un xor avec la représentation binaire sur 32 bits de $2^j$ afin de perturber le $j$-ème bit d'un mot de 32 bits, ce qui revient à réaliser le complément à 1 du bit $j$. Il s'agit du type de perturbation utilisé dans les premières cryptanalyses menées contre la fonction SHA-0 [Wan97, CJ98]. Nous introduisons ici une remarque sur laquelle nous reviendrons de façon détaillée au chapitre 5 section 5-2. Cette façon de générer une perturbation permet de définir une direction pour cette perturbation. Un bit dont la valeur initiale égale à 0 est modifiée en la valeur 1 constitue ce que nous définirons être une perturbation "montante". Réciproquement, un bit égal à 1 modifié en 0 constitue une perturbation "descendante".

Une fois la perturbation initiale introduite, l'objectif est alors d'en neutraliser les effets sur les états des registres. Pour ce faire, on procède à l'introduction de nouvelles perturbations choisies spécifiquement (des corrections) sur les mots de message expansés des pas suivants ($W_i$, $W_{i+1}$, ...).

Nous allons à présent détailler pas à pas un exemple de collision locale. Dans une première approche, nous emploierons la description de la fonction de mise à jour des registres utilisant la notation $(A, B, C, D, E)$.
Nous introduisons la notation $\overset{j}{X}_i$ afin d'indiquer que le bit $j$ du mot de 32 bits $X$ a subi une perturbation au pas $i$. Nous choisissons de définir la numérotation des bits d'un mot en commençant par affecter le numéro 0 au bit de poids faible, puis en incrémentant jusqu'au numéro 31 pour le bit de poids fort. Nous remarquons que les permutations circulaires induisent seulement un déplacement de la position d'un bit perturbé. On a pour $0 \leq n \leq 31$ :

$$X_i \lll n = X_i \ggg (32 - n)$$

et

$$\overset{j}{X}_i \lll n = \overset{j+n}{X_i} \lll n,$$

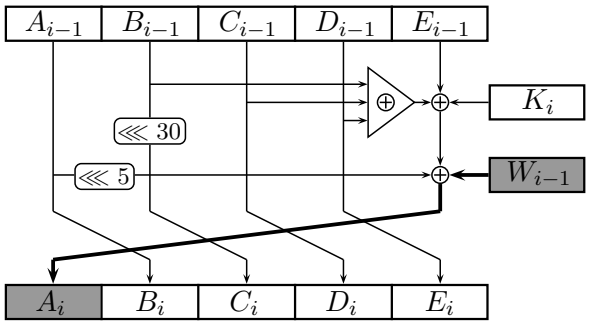
$$\overset{j}{X}_i \ggg n = \overset{j-n}{X_i} \ggg n.$$

De plus, ces permutations n'altèrent pas la direction des perturbations. Cette remarque est vraie pour des perturbations considérées de façon indépendante. Cependant, certaines techniques utilisées lors des cryptanalyses peuvent introduire des perturbations altérant plusieurs bits consécutifs avec des contraintes spécifiques (techniques de compression de bits par exemple). Dans ces cas, les permutations circulaires peuvent interagir avec ces contraintes. Elles doivent alors faire l'objet d'un traitement particulier.

**Pas $i$ :**
Au pas $i$, nous introduisons une perturbation en complémentant à 1 le bit $j$ de $W_{i-1}$. Nous obtenons :

$$\begin{aligned}
&\overset{j}{A}_i = (A_{i-1} \lll 5) \oplus B_{i-1} \oplus C_{i-1} \oplus D_{i-1} \oplus \\
&\qquad E_{i-1} \oplus \overset{j}{W}_{i-1} \oplus K_i, \\
&B_i = A_{i-1}, \\
&C_i = B_{i-1} \lll 30, \\
&D_i = C_{i-1}, \\
&E_i = D_{i-1}.
\end{aligned}$$

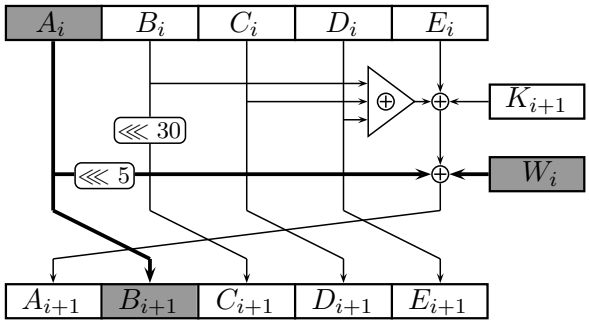
Dans cette version linéaire de la fonction de mise à jour des registres, on constate qu'il résulte de la perturbation du bit $j$ de $W_{i-1}$ la perturbation du seul bit $j$ de $A_i$ qui devient $\overset{j}{A}_i$.




**Pas $i+1$ :**
La perturbation du bit $j$ du registre $A_i$ devrait être répercutée au pas $i+1$ sur le bit $j+5$ de $A_{i+1}$ ainsi que sur le bit $j$ de $B_{i+1}$. Or on obtient :

$$\begin{aligned}
&A_{i+1} = (\overset{j}{A}_i \lll 5) \oplus B_i \oplus C_i \oplus D_i \oplus E_i \oplus \overset{j+5}{W}_i \oplus \\
&\qquad K_{i+1}, \\
&\overset{j}{B}_{i+1} = \overset{j}{A}_i, \\
&C_{i+1} = B_i \lll 30, \\
&D_{i+1} = C_i, \\
&E_{i+1} = D_i.
\end{aligned}$$

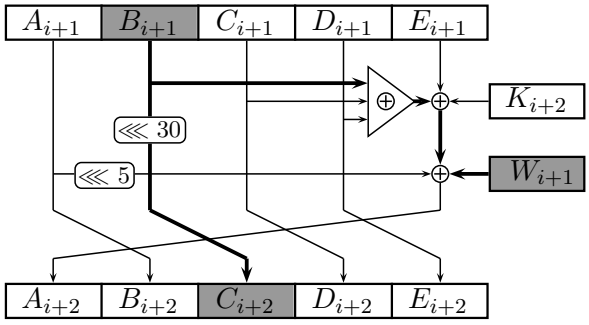
En complémentant à 1 le bit $j+5$ de $W_i$ on corrige la perturbation créée par le terme $(\overset{j}{A}_i \ll 5)$ et donc $A_{i+1}$ n'est pas altéré.




**Pas $i+2$ :**
Au pas $i+2$, la perturbation du bit $j$ de $B_{i+1}$ se transmet au bit $j+30$ de $C_{i+2}$. On a :

$$\begin{aligned}
&A_{i+2} = (A_{i+1} \lll 5) \oplus \overset{j}{B}_{i+1} \oplus C_{i+1} \oplus D_{i+1} \oplus \\
&\qquad E_{i+1} \oplus \overset{j}{W}_{i+1} \oplus K_{i+2}, \\
&B_{i+2} = A_{i+1}, \\
&\overset{j+30}{C}_{i+2} = \overset{j}{B}_{i+1} \lll 30, \\
&D_{i+2} = C_{i+1}, \\
&E_{i+2} = D_{i+1}.
\end{aligned}$$

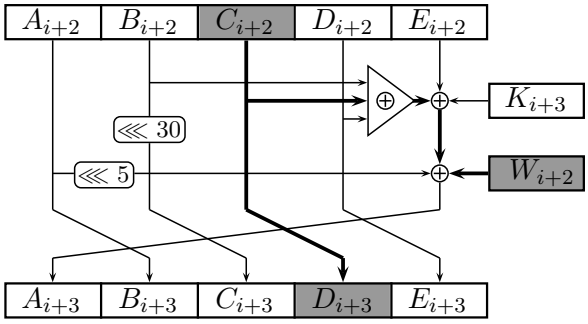
De la même façon qu'au pas $i+1$, on corrige la perturbation introduite par le terme $\overset{j}{B}_{i+1}$ en complémentant à 1 le bit $j$ de $W_{i+1}$.



**Pas $i+3$ :**
La perturbation du bit $j+30$ de $C_{i+2}$ se transmet au bit $j+30$ de $D_{i+3}$. On a :

$$\begin{aligned}
A_{i+3} &= (A_{i+2} \lll 5) \oplus B_{i+2} \oplus \overset{j+30}{C_{i+2}} \oplus D_{i+2} \oplus \\
&\quad E_{i+2} \oplus \overset{j+30}{W_{i+2}} \oplus K_{i+3}, \\
B_{i+3} &= A_{i+2}, \\
C_{i+3} &= B_{i+2} \lll 30, \\
\overset{j+30}{D_{i+3}} &= \overset{j+30}{C_{i+2}}, \\
E_{i+3} &= D_{i+2}.
\end{aligned}$$

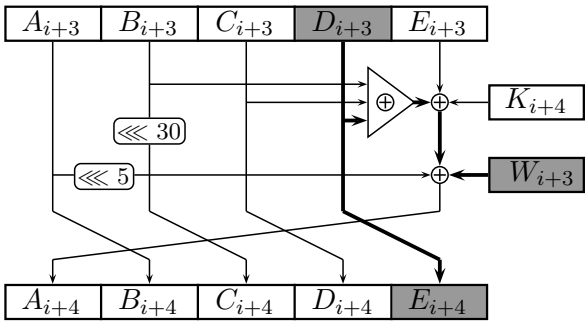
On corrige la perturbation issue du terme $\overset{j}{C_{i+2}}$ en complémentant à 1 le bit $j+30$ de $W_{i+2}$.




**Pas $i+4$ :**
On a :

$$\begin{aligned}
A_{i+4} &= (A_{i+3} \lll 5) \oplus B_{i+3} \oplus C_{i+3} \oplus \overset{j+30}{D_{i+3}} \oplus \\
&\quad E_{i+3} \oplus \overset{j+30}{W_{i+3}} \oplus K_{i+4}, \\
B_{i+4} &= A_{i+3}, \\
C_{i+4} &= B_{i+3} \lll 30, \\
D_{i+4} &= C_{i+3}, \\
\overset{j+30}{E_{i+4}} &= \overset{j+30}{D_{i+3}}.
\end{aligned}$$

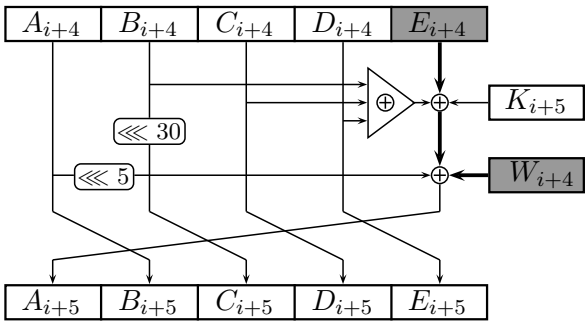
La perturbation du bit $j+30$ de $D_{i+3}$ se transmet au bit $j+30$ de $E_{i+4}$. La correction de $\overset{j+30}{D_{i+3}}$ s'effectue en complémentant à 1 le bit $j+30$ de $W_{i+3}$.



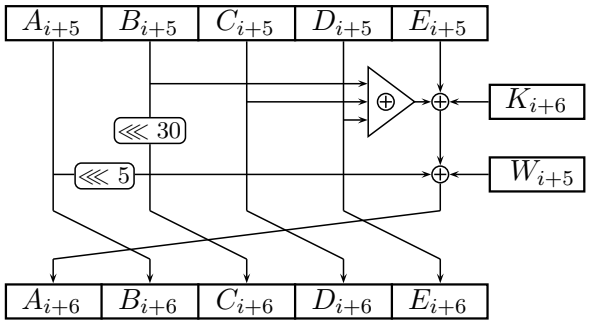

**Pas $i+5$ :**
On a :

$$\begin{aligned}
A_{i+5} &= (A_{i+4} \lll 5) \oplus B_{i+4} \oplus C_{i+4} \oplus D_{i+4} \oplus \\
&\quad \overset{j+30}{E_{i+4}} \oplus \overset{j+30}{W_{i+4}} \oplus K_{i+5}, \\
B_{i+5} &= A_{i+4}, \\
C_{i+5} &= B_{i+4} \lll 30, \\
D_{i+5} &= C_{i+4}, \\
E_{i+5} &= D_{i+4}.
\end{aligned}$$

La dernière correction consiste à complémenter à 1 le bit $j+30$ de $W_{i+4}$.



**Pas $i+6$ :**
Au pas $i+6$, les états des registres sont les mêmes que s'il n'y avait pas eu de perturbation. La perturbation initiale a été entièrement corrigée par les perturbations suivantes.




On peut aussi décrire le déroulement d'une collision locale en fonction du seul registre $A$ :

$$
\begin{aligned}
\overset{\boldsymbol{j}}{A_i} &= (A_{i-1} \lll 5) \oplus A_{i-2} \oplus A_{i-3} \ggg 2 \oplus A_{i-4} \ggg 2 \oplus A_{i-5} \ggg 2 \oplus \ K_i \ \oplus \overset{\boldsymbol{j}}{W_{i-1}} \\
A_{i+1} &= \ \overset{\boldsymbol{j+5}}{(A_i \lll 5)} \ \oplus A_{i-1} \oplus A_{i-2} \ggg 2 \oplus A_{i-3} \ggg 2 \oplus A_{i-4} \ggg 2 \oplus K_{i+1} \oplus \overset{\boldsymbol{j+5}}{W_i} \\
A_{i+2} &= (A_{i+1} \lll 5) \oplus \ \overset{\boldsymbol{j}}{A_i} \ \oplus A_{i-1} \ggg 2 \oplus A_{i-2} \ggg 2 \oplus A_{i-3} \ggg 2 \oplus K_{i+2} \oplus \overset{\boldsymbol{j}}{W_{i+1}} \\
A_{i+3} &= (A_{i+2} \lll 5) \oplus A_{i+1} \oplus \ \overset{\boldsymbol{j+30}}{A_i} \ggg 2 \ \oplus A_{i-1} \ggg 2 \oplus A_{i-2} \ggg 2 \oplus K_{i+3} \oplus \overset{\boldsymbol{j+30}}{W_{i+2}} \\
A_{i+4} &= (A_{i+3} \lll 5) \oplus A_{i+2} \oplus A_{i+1} \ggg 2 \oplus \ \overset{\boldsymbol{j+30}}{A_i} \ggg 2 \ \oplus A_{i-1} \ggg 2 \oplus K_{i+4} \oplus \overset{\boldsymbol{j+30}}{W_{i+3}} \\
A_{i+5} &= (A_{i+4} \lll 5) \oplus A_{i+3} \oplus A_{i+2} \ggg 2 \oplus A_{i+1} \ggg 2 \oplus \ \overset{\boldsymbol{j+30}}{A_i} \ggg 2 \ \oplus K_{i+5} \oplus \overset{\boldsymbol{j+30}}{W_{i+4}} \\
A_{i+6} &= (A_{i+5} \lll 5) \oplus A_{i+4} \oplus A_{i+3} \ggg 2 \oplus A_{i+2} \ggg 2 \oplus A_{i+1} \ggg 2 \oplus K_{i+6} \oplus \ W_{i+5}
\end{aligned}
$$

La séquence de perturbations qui apparaissent dans les équations des états de ce registre constitue ce que l'on nomme un chemin différentiel.

#### 4-2.1.1 Conclusion

Nous venons donc de décrire une perturbation et une séquence de cinq corrections appropriées qui forment une collision locale pour une version linéaire de la fonction de mise à jour des registres commune à SHA-0 et SHA-1. Les séquences de mots de message expansé :

$$(W_{i-1}, W_i, W_{i+1}, W_{i+2}, W_{i+3}, W_{i+4})$$

et

$$(\overset{j}{W_{i-1}}, \overset{j+5}{W_i}, \overset{j}{W_{i+1}}, \overset{j+30}{W_{i+2}}, \overset{j+30}{W_{i+3}}, \overset{j+30}{W_{i+4}})$$

conduisent toutes deux à un même état pour les cinq registres, ce qui constitue bien une collision.

Une autre façon de considérer une collision locale consiste à dire que pour une version linéaire de la fonction de mise à jour des registres réduite à 6 pas, nous avons construit une différentielle

$$\Delta =< 2^j, 2^{j+5}, 2^j, 2^{j+30}, 2^{j+30}, 2^{j+30} >$$

telle que le couple de messages $W =< W_0, W_1, W_2, W_3, W_4, W_5 >$ et $W' = W \oplus \Delta$ forme une collision.

De plus, pour cette version linéaire de la fonction de mise à jour des registres, le modèle se comporte de façon idéale. Pour tout couple $(M, M')$, nous obtiendrons une collision locale avec une probabilité égale à 1. Ce qui ne sera plus le cas dès lors que les parties non-linéaires de la fonction de mise à jour des registres, l'addition modulo $2^{32}$ ainsi que les fonctions $IF$ et $MAJ$, interviendrons.

### 4-2.2 Fonction de mise à jour des registres standard

Nous venons de vérifier que le modèle de collision locale fonctionne parfaitement pour une version linéaire de la fonction de mise à jour des registres commune au deux fonctions SHA-0 et SHA-1. À présent, il convient de s'intéresser aux éléments non-linéaires de cette fonction qui sont susceptibles d'altérer ou d'invalider ce modèle : l'addition modulo $2^{32}$ et les fonctions de ronde $f_i$.

#### 4-2.2.1 Addition modulo $2^{32}$

L'un des éléments qui induisent une différence de comportement par rapport à la version totalement linéaire de la fonction de mise à jour des registres réside dans l'addition modulo $2^{32}$. En effet, cette addition peut entraîner l'apparition de retenues, et donc propager une perturbation sur un bit d'indice $j$ vers d'autres bits d'indices supérieurs. Si on considère la perturbation initiale et les cinq corrections qui en découlent, chacune de ces manipulations est susceptible lors du calcul de l'état du registre d'amener la création d'une retenue. Cependant, la notion de direction de perturbation que nous avons brièvement décrite peut être utilisée pour éviter la création de certaines retenues en introduisant un mécanisme de compensation des perturbations. De plus, une façon directe de diminuer l'impact des retenues consiste à choisir judicieusement le bit $j$ que l'on va altérer. En effet, on peut noter que trois des corrections ont lieu sur le bit $j+30$ et en choisissant $j = 1$, ces corrections s'effectueront sur le bit 31 qui est le bit le plus significatif. Or s'agissant d'une addition modulo $2^{32}$, une éventuelle retenue apparaissant au niveau du bit numéro 31 est ignorée. C'est pourquoi, les auteurs des premières attaques différentielles contre les fonctions SHA-0 et SHA-1 ont choisi de privilégier comme séquence de perturbation/corrections pour une collision locale initiée au pas $i$ : $(\overset{1}{W}_{i-1}, \overset{6}{W}_i, \overset{1}{W}_{i+1}, \overset{31}{W}_{i+2}, \overset{31}{W}_{i+3}, \overset{31}{W}_{i+4})$. Cependant, de possibles retenues peuvent toujours apparaître pour les positions de bit 1 et 6.

#### 4-2.2.2 Fonctions de ronde $IF$ et $MAJ$

Considérons à présent, l'impact des fonctions de rondes. Pour une version de la fonction de mise à jour des registres où l'addition modulo $2^{32}$ est remplacée par l'addition bit à bit modulo 2 sur des mots de 32 bits, et où les fonctions de ronde sont inchangées, les équations sur les états des registres s'écrivent :

$A_i = (A_{i-1} \lll 5) \oplus f_i(B_{i-1}, C_{i-1}, D_{i-1}) \oplus E_{i-1} \oplus W_{i-1} \oplus K_i,$

$B_i = A_{i-1},$

$C_i = B_{i-1} \lll 30,$

$D_i = C_{i-1},$

$E_i = D_{i-1}$.

avec

$f_i = IF$ pour $i \in \{1, \ldots, 20\}$ (ronde 1),
$f_i = XOR$ pour $i \in \{21, \ldots, 40\} \cup \{61, \ldots, 80\}$ (rondes 2 et 4),
$f_i = MAJ$ pour $i \in \{41, \ldots, 60\}$ (ronde 3).

On peut tout d'abord remarquer que pour les rondes 2 et 4, la fonction $f_i$ utilisée étant la fonction $XOR$, on se ramène à la version linéaire de la fonction de mise à jour des registres. En conséquence, pour la version standard de la fonction de mise à jour des registres, le comportement que nous avons détaillé ne sera affecté que par l'addition modulo $2^{32}$ pour ces deux rondes. Pour les rondes 1 et 3, les fonctions utilisées sont respectivement la fonction $IF$ et la fonction $MAJ$ et ces deux fonctions ne sont pas linéaires. Elles possèdent la particularité de pouvoir "absorber" une perturbation : une perturbation présente sur un bit d'un de leurs arguments peut ne pas être répercutée sur la sortie de la fonction. Ce comportement est susceptible d'altérer le chemin différentiel de telle sorte qu'il soit impossible d'obtenir une collision locale : nous parlerons alors de chemin différentiel impossible. De plus, si l'on considère la direction d'une perturbation, chacune de ces trois fonctions est susceptible de modifier la direction d'une perturbation, ce qui peut invalider les mécanismes de compensation des perturbations.

## 4-3 Chemin différentiel

Rechercher une collision pour la fonction de compression de SHA-0 ou de SHA-1, revient à construire un chemin différentiel pour les 80 pas de la fonction de compression. Ce chemin différentiel est constitué par les perturbations issues des séquences de collisions locales introduites au moyen des mots de message expansés, ces collisions locales pouvant de plus se chevaucher. Pour construire un chemin différentiel, il faut identifier un ensemble approprié de points de départ pour chacune des collisions locales. Ce qui nous conduit à introduire les notions de vecteur de perturbations, de masque de perturbations et de caractéristique linéaire.

### 4-3.1 Vecteur de perturbations

Lors de l'étude de la version linéaire de mise à jour des registres, nous avons posé comme hypothèse que nous pouvions intervenir sur les mots de message étendu $W_i$ indépendamment du processus d'expansion dont ils sont issus. Nous devons à présent prendre en compte le fait que les mots $W_i$ sont issus d'un processus d'expansion. En effet, la relation de récurrence de SHA-0 (3-1) (respectivement SHA-1 (3-2)) répercutera les perturbations appliquées sur les 16 premiers mots $W_0, \ldots, W_{15}$ à l'ensemble des mots expansés.

Pour définir les points de départ de chacune des collisions locales, on définit un vecteur représentant les indices des pas où seront initiées les perturbations. Ce vecteur, appelé vecteur de perturbations (*Disturbance Vector*), est constitué de 80 mots de 32 bits dont les coordonnées non nulles indiquent les positions où une collision locale est initiée. De plus afin de se conformer au processus d'expansion, on impose que les coordonnées de ce vecteur vérifient la relation de récurrence (3-1) pour SHA-0 ou (3-2) pour SHA-1.

L'équation récursive définissant l'expansion utilisée pour SHA-0 n'est composée que d'additions modulo 2 sur 32 bits. Il en découle que chacune des coordonnées des mots expansés peuvent être obtenues parallèlement. En conséquence, on peut choisir de positionner toutes les coordonnées non nulles du vecteur de perturbation sur une même position de bit $j$. Pour la fonction SHA-0, un tel vecteur se définit alors de la façon suivante :

$$V = (V_0, \ldots, V_{79}),$$
$$\text{avec } V_i = \begin{cases} (2^j)_x \text{ si et seulement si on initie une perturbation locale sur} \\ \qquad \text{le bit } j \text{ au pas } i+1, \\ 0_x \text{ sinon.} \end{cases}$$

Dans le cas de la fonction SHA-1, la permutation circulaire induit une diffusion des perturbations sur des positions de bit différentes. Nous pouvons toujours définir notre vecteur de perturbations, mais il n'est plus possible de limiter les coordonnées non nulles à une seule projection. Nous entrevoyons ici, la différence fondamentale du point de vue de la cryptanalyse entre SHA-0 et SHA-1. Nous reviendrons plus largement sur ce point au chapitre 5 section 5-3.

On peut dès à présent faire la remarque suivante : dans le modèle de collision locale, une perturbation démarrant au pas $i$ ne peut être totalement corrigée avant le pas $i+6$. En conséquence, afin d'obtenir une collision pour l'ensemble de la fonction de compression, toutes les perturbations se doivent d'avoir été corrigées au pas 80, ce qui impose pour le vecteur de perturbations les conditions supplémentaires :

$$V_{79} = V_{78} = V_{77} = V_{76} = V_{75} = 0_x.$$

Cette condition n'est nécessaire que dans le cas où l'on recherche une collision sur une seule itération de la fonction de compression. Si l'on souhaite aboutir à une quasi-collision, par exemple dans le but d'utiliser la technique des collisions multiples, on peut s'affranchir de cette contrainte.

### 4-3.2 Masque de perturbations

Le vecteur de perturbations représente les points de départ des collisions locales. On souhaite, à partir de ce vecteur, définir le masque de perturbations $P$, constitué de 80 mots de 32 bits, à appliquer au message expansé $W$ pour construire le message $W' = W \oplus P$. Le masque de perturbations représente les perturbations initiales ainsi que les séquences de corrections.

Cependant, pour définir de façon générale ce masque, il faut étendre le vecteur de perturbations avec cinq nouvelles coordonnées $V_{-1}$, $V_{-2}$, $V_{-3}$, $V_{-4}$ et $V_{-5}$, le cas $V_{-1} = V_{-2} = V_{-3} = V_{-4} = V_{-5} = 0_x$ signifiant qu'il n'y a pas, sur les cinq premiers mots $W_i$, d'altération due à des corrections. On peut alors définir le masque de perturbations de la façon suivante :

$$P = (P_0, \ldots, P_{79}),$$
$$\text{avec pour } i = 0, \ldots, 79,$$
$$P_i = V_i \oplus (V_{i-1} \lll 5) \oplus V_{i-2} \oplus (V_{i-3} \ggg 2) \oplus (V_{i-4} \ggg 2) \oplus (V_{i-5} \ggg 2).$$

Le premier terme $V_i$ correspond aux perturbations initiales et les termes suivants $(V_{i-1} \lll 5)$, $V_{i-2}$, $(V_{i-3} \ggg 2)$, $(V_{i-4} \ggg 2)$ et $(V_{i-5} \ggg 2)$, aux séquences de corrections à appliquer.

Pour s'assurer que le message $W'$ puisse appartenir à l'ensemble image du processus d'expansion, on impose que le masque de perturbations vérifie une relation de récurrence identique :

$$\text{SHA-0} : P_i = P_{i-3} \oplus P_{i-8} \oplus P_{i-14} \oplus P_{i-16} \text{ avec } 11 \leq i \leq 79,$$
$$\text{SHA-1} : P_i = (P_{i-3} \oplus P_{i-8} \oplus P_{i-14} \oplus P_{i-16}) \lll 1 \text{ avec } 11 \leq i \leq 79.$$

On remarque que pour les cinq indices négatifs il s'agit de perturbations virtuelles. Bien qu'aucune perturbation ne puisse réellement exister avant le premier pas de la fonction de compression, on prend en compte les indices négatifs pour la construction du masque de perturbations. Des coordonnées non nulles sur les cinq premiers $P_i$ seront alors interprétées comme des corrections de perturbations virtuelles. Dans la littérature consacrée, ce phénomène est dénommé collisions locales tronquées (*Truncated Local Collisions*). Compte tenu de cette remarque, il est facile de voir que la donnée des coordonnées $(V_0, \ldots, V_{15})$ définit complètement le masque de perturbations.

### 4-3.3 Caractéristique linéaire (*Linear Characteristic*).

La notion de caractéristique linéaire a été introduite par De Cannière et Rechberger dans leur article de 2006 [DCR06]. Cette notion est définie par opposition à une caractéristique non-linéaire que nous détaillerons au chapitre 5 section 5-4. La dénomination linéaire découle du fait que cette caractéristique possède une probabilité de succès égale à 1 pour la version totalement linéaire de la fonction de mise à jour des registres. Pour cette version, il est facile de voir que le chemin différentiel suivi par les différents états du registre $A$ est le même que celui décrit par le vecteur de perturbation. Il correspond au cas où toutes les collisions locales introduites se comportent idéalement.

Cependant, la caractéristique linéaire décrite dans l'article De Cannière et Rechberger inclut de plus, des informations supplémentaires liées à la cryptanalyse. Ces informations se présentent sous la forme de conditions imposées à certains bits des mots de message expansés. Par extension, on utilisera le terme de caractéristique linéaire pour décrire selon le contexte, le vecteur de perturbations et/ou le chemin différentiel.

## 4-4 Mise en oeuvre du modèle

Nous avons souligné dans les sections précédentes que le comportement d'une collision locale était fortement influencé par d'une part l'addition modulo $2^{32}$ et d'autre part par les fonctions de ronde $IF$ et $MAJ$. Nous allons à présent nous intéresser plus particulièrement au traitement de ces fonctions en différenciant deux approches que nous qualifierons d'approche probabiliste et d'approche déterministe.

### 4-4.1 Approche probabiliste

C'est celle qui fût utilisée dans les premières cryptanalyses de SHA-0, [Wan97], [CJ98] puis [BC04]. Elle consiste à évaluer la probabilité que les fonctions de ronde $IF$ et $MAJ$ aient un comportement identique à la fonction $XOR$.

Si nous choisissons par exemple de définir la probabilité que la valeur de sortie de la fonction $IF$ reste inchangée lorsque ses arguments sont modifiés, par rapport à cette même probabilité pour la fonction $XOR$, le comportement de la fonction $IF$ peut s'exprimer de la façon suivante :

$$Pr[IF(X,Y,Z) = IF(\overset{j}{X},Y,Z)] = 1/2,$$
$$Pr[IF(X,Y,Z) = IF(X,\overset{j}{Y},Z)] = 1/2,$$
$$Pr[IF(X,Y,Z) = IF(X,Y,\overset{j}{Z})] = 1/2,$$
$$Pr[IF(X,Y,Z) = IF(\overset{j}{X},\overset{j}{Y},Z)] = 1/2,$$
$$Pr[IF(X,Y,Z) = IF(\overset{j}{X},Y,\overset{j}{Z})] = 1/2,$$
$$Pr[IF(X,Y,Z) = IF(X,\overset{j}{Y},\overset{j}{Z})] = 0,$$
$$Pr[IF(X,Y,Z) = IF(\overset{j}{X},\overset{j}{Y},\overset{j}{Z})] = 1/2.$$

Et le comportement de la fonction $XOR$, quant à lui s'écrit :

$$Pr[XOR(X,Y,Z) = XOR(\overset{j}{X},Y,Z)] = 0,$$
$$Pr[XOR(X,Y,Z) = XOR(X,\overset{j}{Y},Z)] = 0,$$
$$Pr[XOR(X,Y,Z) = XOR(X,Y,\overset{j}{Z})] = 0,$$
$$Pr[XOR(X,Y,Z) = XOR(\overset{j}{X},\overset{j}{Y},Z)] = 1,$$
$$Pr[XOR(X,Y,Z) = XOR(\overset{j}{X},Y,\overset{j}{Z})] = 1,$$
$$Pr[XOR(X,Y,Z) = XOR(X,\overset{j}{Y},\overset{j}{Z})] = 1,$$
$$Pr[XOR(X,Y,Z) = XOR(\overset{j}{X},\overset{j}{Y},\overset{j}{Z})] = 0.$$

On remarque que ces probabilités recouvrent tous les cas possibles car nous avons affaire à des fonctions qui n'entrelacent pas les bits (fonctions bit à bit). La fonction $IF$ se comporte comme la fonction $XOR$ avec une probabilité égale à $1/2$ sauf pour le cas où $X$ et $Y$ sont modifiés simultanément sur la même position de bit. Alors, la fonction $IF$ ne se comportera jamais comme la fonction $XOR$, ce qui constitue un exemple de chemin différentiel impossible.

Le principe d'une attaque différentielle fondée sur une approche probabiliste consistera donc à éviter les chemins différentiels impossibles et à évaluer la probabilité de bon comportement pour chacune des fonctions $IF$ et $MAJ$, et pour chaque collision locale définie par le vecteur de perturbations. Une fois ce travail réalisé, on peut évaluer la complexité de l'attaque comme une fonction de l'inverse du produit de toutes ces probabilités.

### 4-4.2 Approche déterministe

Dans cette approche, on ne cherche plus à évaluer la probabilité que les fonctions $IF$ et $MAJ$ se comportent comme la fonction $XOR$, mais à déterminer des conditions suffisantes sur leurs arguments pour prévoir le comportement de leur sortie. Cette approche est celle initiée par Wang dans [Wan98].

Prenons là encore la fonction $IF$ comme exemple. La fonction $IF$ prend comme arguments trois mots de 32 bits et retourne un mot de 32 bits :

$$\forall X, Y \text{ et } Z \in \{0,1\}^{32} \quad IF(X,Y,Z) = (X \wedge Y) \vee (\neg X \wedge Z).$$

Cependant comme nous l'avons déjà remarqué, les opérations utilisées dans le calcul de $IF(X,Y,Z)$ sont des opérations bit à bit, on peut donc considérer la fonction $IF$ comme la concaténation de 32 instances d'une fonction élémentaire $if$ :

$$\forall x, y \text{ et } z \in \{0,1\} \quad if(x,y,z) = (x \wedge y) \vee (\neg x \wedge z).$$

On a alors, pour tout $X$, $Y$ et $Z$ dans $\{0,1\}^{32}$ tels que $X = (x_0, x_1, \ldots, x_{31})$, $Y = (y_0, y_1, \ldots, y_{31})$ et $Z = (z_0, z_1, \ldots, z_{31})$ :

$$IF(X,Y,Z) =< if(x_0,y_0,z_0), if(x_1,y_1,z_1), \ldots, if(x_{31},y_{31},z_{31}) > .$$

On peut appliquer le même principe pour les fonctions $XOR$ et $MAJ$.

Voici alors une description partielle du comportement de la fonction élémentaire $if$ :

$$\begin{aligned}
if(x,y,z) = \;& if(\neg x,y,z) && \text{si} \quad y = z,\\
= \;& \neg if(\neg x,y,z) && \text{si} \quad y = \neg z,\\
= \;& if(x,\neg y,z) && \text{si} \quad x = 0,\\
= \;& \neg if(x,\neg y,z) && \text{si} \quad x = 1,\\
= \;& if(x,y,\neg z) && \text{si} \quad x = 1,\\
= \;& \neg if(x,y,\neg z) && \text{si} \quad x = 0,\\
\neq \;& if(x,\neg y,\neg z) && \text{dans tous les cas.}
\end{aligned}$$

Le principe d'une attaque différentielle fondée sur l'approche déterministe consiste donc à énumérer toutes les conditions suffisantes pour que les fonctions de rondes aient un comportement en adéquation avec le modèle de collision locale. On évalue alors la complexité de l'attaque comme une fonction du nombre total de conditions à vérifier.

### 4-4.3 Hypothèse d'indépendance

La nature de ces deux approches tient au point de vue choisi pour aborder le problème. La première approche résulte d'une démarche idéaliste alors que la seconde résulte d'une démarche empirique. Cependant, ces deux approches reposent sur la même hypothèse d'indépendance : l'entrelacement de différentes collisions locales ne génère pas d'effets de bord autres que ceux pris en compte dans les deux approches. Or différents travaux, ainsi que les expérimentations réalisées durant la dernière année de cette thèse ont démontré qu'il existait des interactions supplémentaires. Nous reviendrons plus largement sur ce point au chapitre 8.

### 4-4.4 Évaluation du comportement d'une collision locale

L'addition modulo $2^{32}$ ainsi que les parties non linéaires de la fonction de mise à jour des registres influent sur le comportement du modèle de la collision locale. Dans certain cas, on pourra même aboutir à un chemin différentiel impossible. Nous détaillons à présent un exemple d'évaluation du comportement d'une collision locale selon les deux approches que nous venons d'introduire et en prenant comme hypothèse que la collision locale que nous considérons se "déroule" au sein de la ronde 1 ($IF$).

Si nous appliquons la caractéristique linéaire d'une collision locale à la version standard de la fonction de mise à jour des registres, un comportement idéal pourrait être décrit de la façon suivante :

$$
\begin{aligned}
A_{i+1} &= (A_i \lll 5) + IF(A_{i-1}, A_{i-2} \ggg 2, A_{i-3} \ggg 2) + A_{i-4} \ggg 2 + K_i + W_i^{\boldsymbol{j}} \\
A_{i+2} &= (A_{i+1} \lll 5) + IF(A_i^{\boldsymbol{j+5}}, A_{i-1} \ggg 2, A_{i-2} \ggg 2) + A_{i-3} \ggg 2 + K_{i+1} + W_{i+1}^{\boldsymbol{j+5}} \\
A_{i+3} &= (A_{i+2} \lll 5) + IF(A_{i+1}^{\boldsymbol{j}}, A_i \ggg 2, A_{i-1} \ggg 2) + A_{i-2} \ggg 2 + K_{i+2} + W_{i+2}^{\boldsymbol{j}} \\
A_{i+4} &= (A_{i+3} \lll 5) + IF(A_{i+2}, A_{i+1}^{\boldsymbol{j+30}} \ggg 2, A_i \ggg 2) + A_{i-1} \ggg 2 + K_{i+3} + W_{i+3}^{\boldsymbol{j+30}} \\
A_{i+5} &= (A_{i+4} \lll 5) + IF(A_{i+3}, A_{i+2} \ggg 2, A_{i+1}^{\boldsymbol{j+30}} \ggg 2) + A_i \ggg 2 + K_{i+4} + W_{i+4}^{\boldsymbol{j+30}} \\
A_{i+6} &= (A_{i+5} \lll 5) + IF(A_{i+4}, A_{i+3} \ggg 2, A_{i+2} \ggg 2) + A_{i+1}^{\boldsymbol{j+30}} \ggg 2 + K_{i+5} + W_{i+5}^{\boldsymbol{j+30}}
\end{aligned}
$$

Évaluons alors le comportement de cette collision locale :

1. Pour que la première des 6 équations que nous venons d'écrire soit vérifiée, il suffit que la transmission de la perturbation introduite au mot $W_i$ se fasse sans retenue. Cette condition est vérifiée avec une probabilité égale à 1/2 si l'on considère l'angle probabiliste et correspond à la condition $a_{i+1,j} = w_{i,j}$ selon le point de vue déterministe.
2. Pour la deuxième équation, on remarque que si la première équation est vérifiée, la seconde le sera également si l'on prend la précaution d'imposer que la correction introduite sur le bit $j+5$ de $W_{i+1}$ soit de direction opposée à la perturbation introduite sur le bit $j$ de $W_i$, ce que l'on peut représenter sous la condition $w_{i+1,j+5} = \neg w_{i,j}$. À noter que cette condition est sous le contrôle de l'attaquant puisque elle ne dépend que des mots de message expansés ; ce type de condition n'entre donc pas en compte dans l'évaluation du comportement d'une collision locale. Il faudra cependant prendre garde à la vérification des conditions de ce type lors de la construction effective d'une attaque.
3. La troisième équation voit l'apparition de l'action de la fonction de ronde. Pour que cette équation soit vraie, il suffit que dans un premier temps la perturbation présente sur le registre $A_{i+1}$ ne soit pas absorbée par la fonction de ronde. Puis dans un second temps, la perturbation issue de la sortie de la fonction $IF$ et la correction introduite par le mot $W_{i+2}$ doivent se compenser. Si nous nous ramenons aux descriptions du comportement de la fonction $IF$ données aux sections précédentes, nous constatons que dans le cas qui nous intéresse (il existe une perturbation sur un bit du premier argument de la fonction), d'un point de vue probabiliste la fonction se comportera de façon identique à la

fonction $XOR$ avec une probabilité égale à 1/2. Du point de vue déterministe, si la condition $a_{i,j+30} \neq a_{i-1,j+30}$ est vérifiée alors la perturbation ne sera pas absorbée. Cependant, la fonction $IF$ peut préserver ou inverser la direction de la perturbation ce qui peut compromettre la compensation des perturbations, même si une contrainte consistant à imposer que la correction introduite sur le bit $j$ de $W_{i+2}$ soit de direction opposée à la direction de la perturbation introduite sur le bit $j$ de $W_i$ est utilisée. Il existe alors deux solutions pour traiter le problème :

- ne pas imposer de contrainte sur la direction de la perturbation du bit $j$ de $W_{i+2}$ et considérer une probabilité supplémentaire de 1/2 pour que la fonction $IF$ préserve la direction,
- imposer la contrainte $w_{i+2,j} = \neg w_{i,j}$ et transformer la condition déterministe en $a_{i,j+30} \neq a_{i-1,j+30} = 1$.

Finalement, la troisième équation est vérifiée avec une probabilité égale à 1/4 ou si la double condition $a_{i,j+30} \neq a_{i-1,j+30} = 1$ est vérifiée.

4. Pour la quatrième et la cinquième équation, le comportement de la fonction $IF$ change : une perturbation sur le deuxième ou le troisième argument de la fonction est soit absorbée, soit préservée avec la bonne direction. Il en découle, qu'en imposant les conditions $w_{i+3,j+30} = w_{i+4,j+30} = \neg w_{i,j}$, ces équations seront valides avec une probabilité 1/2 (absorption ou non) ou si les équations $a_{i+1,j+30} = 1$ et $a_{i+2,j+30} = 0$ sont vérifiées.
5. Pour la dernière équation, la fonction de ronde n'intervient plus, il y aura donc compensation dès lors que l'on impose $w_{i+5,j+30} = \neg w_{i,j}$.

Nous pouvons mener les mêmes analyses pour chacune des 32 positions de bit possibles et chacune des 3 fonctions Booléennes. Nous présentons dans le tableau 4-1 les probabilités de bon comportement (le nombre de conditions à vérifier) d'une collision locale isolée qui se déroule intégralement dans une même ronde.

| Type | ----------o------ | | | |
|---|---|---|---|---|
| | 0 | 1 | 2, ..., 30 | 31 |
| $IF$ | 5 (5) | 5 (5) | 5 (5) | 4 (4) |
| $XOR$ | 4 (4) | 2 (2) | 4 (4) | 3 (3) |
| $MAJ$ | 4 (4) | 2 (2) | 4 (4) | 4 (4) |

TAB. 4-1 – Probabilités théoriques de bon comportement (nombre de conditions à vérifier) pour une collision locale isolée commençant et se terminant au sein d'une même ronde. La figure utilisée à droite de la notation "Type" représente une collision locale isolée. La numérotation 0, 1, 2, ...30, 31 indique la position à laquelle la perturbation initiale est injectée. Les probabilités sont données sous la forme de l'opposé de leur logarithme en base 2. Les chiffres entre parenthèses donnent le nombre de conditions à vérifier.

On peut remarquer que les approches que nous avons choisi de qualifier de probabiliste et déterministe se rejoignent, les conditions utilisées dans l'approche déterministe possédant une certaine probabilité d'être vérifiées, égale à 1/2 dans l'hypothèse où les bits des différents états des registres sont considérés comme étant aléatoires.

Ces probabilités et conditions ont servi de base à l'évaluation de la complexité des premières attaques menées contre SHA-0. Elles reposent sur plusieurs hypothèses : l'évaluation globale du comportement de la collision locale repose sur les différentes évaluations menées indépendamment équation par équation et les bits des différents états des registres sont considérés comme étant aléatoires. C'est pourquoi, nous les qualifions de probabilités théoriques.

Mendel *et al.* [MPR06] ont légèrement amélioré ces probabilités en prenant en compte de façon détaillée l'impact des retenues. Cependant, les éventuels effets de bords liés aux enchevêtrements de collisions locales n'avaient été que partiellement pris en compte jusqu'à récemment. Nous avons montré que certaines hypothèses d'indépendance sont remises en cause en pratique dès lors que plusieurs collisions locales cohabitent sur un ou plusieurs pas. Ce travail, abordé au chapitre 8, doit paraître dans le journal Design, Codes and Cryptographie [Man10].

# 5

# Cryptanalyses pratiques

**Sommaire**



## 5-1 Introduction

Pour mener à bien une cryptanalyse différentielle des fonctions SHA-0 ou SHA-1 dans le but de produire une collision, un certain nombres d'outils doivent être mis en oeuvre.

La première étape du processus consiste à choisir un opérateur de différence. Les différents progrès obtenus sont fondamentalement dus à la mise en oeuvre d'opérateurs de plus en plus pertinents ainsi qu'à l'évolution de la façon de les représenter.

Nous présentons ces opérateurs section 5-2. L'étape suivante consiste à trouver une caractéristique linéaire susceptible de conduire de façon efficace à une collision. La recherche d'une telle caractéristique se fonde sur l'obtention de bons vecteurs de perturbations. Cet aspect est une première fois abordé section 5-3 puis développé dans le chapitre 7. Il s'avère cependant, que les vecteurs de perturbations les plus prometteurs induisent des collisions tronquées. Un nouvel outil s'est alors révélé nécessaire afin d'adapter la cryptanalyse à ces vecteurs. Pour ce faire, la notion de caractéristique non-linéaire a été introduite. Nous présentons cette notion section 5-4. Avec le développement de ces outils, la technique des blocs multiples a pu être utilisée afin d'améliorer l'attaque. Nous décrivons deux cas d'utilisation de cette technique dans la section 5-5.

Une fois l'opérateur de différence défini, la caractéristique linéaire trouvée et la caractéristique non-linéaire construite, la dernière étape consiste à rechercher un couple de messages formant une collision. Des techniques d'accélération de recherche ont été développées afin de mener efficacement cette recherche, nous les décrivons section 5-6.

## 5-2 Opérateur de différence

Les cryptanalyses mises en oeuvre contre SHA-0 et SHA-1 s'appuient sur l'utilisation des collisions locales. Le travail du cryptanalyste consiste donc à définir un chemin différentiel et à contrôler le comportement de se chemin. La première étape de cette tâche consiste à définir un opérateur de différence. Le premier opérateur utilisé fût le *ou exclusif*, cependant au fur et à mesure de l'évolution des cryptanalyses de nouveaux opérateurs ont été proposés. Nous décrivons dans cette section ces différents opérateurs.

Nous reprenons partiellement les notations employées dans la thèse de Thomas Peyrin [Pey08].

### 5-2.1 Différence binaire

La différence binaire repose sur l'utilisation du *ou exclusif* comme opérateur de différence. Elle correspond aux premières cryptanalyses menées contre la fonction SHA-0 [Wan97, CJ98]. Nous pouvons la définir de la façon suivante :

**Définition 5.1 (Différence binaire)**
*Soient $X$ et $Y$ deux mots binaires de taille $n$, $(X, Y) \in \{0,1\}^n \times \{0,1\}^n$, leur différence binaire $\Delta^{\oplus}(X, Y) \in \{0,1\}^n$ vaut :*

$$\Delta^{\oplus}(X, Y) = X \oplus Y.$$

La différence binaire est une différence simple qui fournit peu d'information. Cependant dans leur article, Chabaud et Joux ont introduit la notion de "direction" pour une différence, celle-ci peut être "montante" ou "descendante". Pour un couple de message expansés $(W, W')$ avec $W' = W \oplus 2^j$, la différence est dite montante si le bit $j$ de $W$ est égal à 0, descendante sinon. Ces deux notions fournissent une information supplémentaire au cryptanalyste et préfigurent la différence binaire signée.

### 5-2.2 Différence binaire signée

Cet opérateur a été introduit pour la cryptanalyse de SHA-0 par Wang [Wan98], puis mis en pratique avec succès dans [WYY05d]. La différence utilisée n'est plus simplement le *ou exclusif* mais une différence fondée sur la soustraction arithmétique modulo 2. L'utilisation d'un signe permet d'incorporer de façon explicite l'information sur la direction de la différence décrite par Chabaud et Joux. Afin de définir la différence binaire signée, nous allons tout d'abord introduire ce que nous nommerons différence binaire signée élémentaire :

**Définition 5.2 (Différence binaire signée élémentaire)**
*Soient deux valeurs binaires $x$ et $y$, leur différence binaire signée élémentaire $\delta \in \{-1, 0, +1\}$ vaut :*

$$\delta(x, y) = y - x.$$

Dans le cadre d'une cryptanalyse différentielle les deux valeurs à considérer sont souvent notées $x$ et $x'$. Nous emploierons dès lors la notation $\delta_x$ pour désigner $\delta(x, x')$. Nous pouvons à présent définir la différence binaire signée de la façon suivante :

**Définition 5.3 (Différence binaire signée)**
*Soient $X$ et $Y$ deux mots binaires de taille $n$ bits, $X = x_0, \ldots, x_{n-1}$ et $Y = y_0, \ldots, y_{n-1}$. Leur différence binaire signée $\Delta^{\pm}(X, Y)$ est le vecteur définit par :*

$$\Delta^{\pm}(X, Y) = (\delta(x_0, y_0), \ldots, \delta(x_{n-1}, y_{n-1})) = (y_0 - x_0, \ldots, y_{n-1} - x_{n-1}).$$

La différence binaire signée permet de tirer pleinement partie du comportement de la fonction $IF$. On peut exprimer le comportement de la fonction élémentaire $if$ en fonction de la différence binaire signée élémentaire de sa sortie $\delta_{if}(x, y, z)$ et des différences binaires signées élémentaires de ses entrées $\delta_x$, $\delta_y$ et $\delta_z$, avec pour tout $(x, x') \in \{0, 1\} \times \{0, 1\}$, $\delta_x \in \{-1, 0, +1\}$. Si on considère une différence binaire signée, la fonction $IF$ est susceptible d'en préserver ou d'en changer la direction (le signe), voire même d'absorber cette différence. Par exemple, pour le cas où $\delta_x \neq 0$ et $\delta_y = \delta_z = 0$, on a :

$$\begin{aligned}
\delta_{if}(x, y, z) &= +\delta_x \quad \text{si} \quad y = \neg z = 1, \\
\delta_{if}(x, y, z) &= -\delta_x \quad \text{si} \quad y = \neg z = 0, \\
\delta_{if}(x, y, z) &= 0 \quad \text{si} \quad y = z.
\end{aligned}$$

Cet exemple illustre clairement la capacité de la fonction $if$ à absorber une différence ($\delta_{if}(x, y, z) = 0$ si $y = z$), à en préserver la direction ($\delta_{if}(x, y, z) = +\delta_x$ si $y = \neg z = 1$) ou à en inverser la direction ($\delta_{if}(x, y, z) = -\delta_x$ si $y = \neg z = 0$).

La différence binaire signée s'intègre donc parfaitement à l'approche déterministe en intégrant l'information supplémentaire liée au signe de la différentielle introduite. Le cryptanalyste possède avec cet outil un contrôle relatif sur le déroulement du chemin différentiel.

| $(x, x')$ | $(0,0)$ | $(1,0)$ | $(0,1)$ | $(1,1)$ |
|---|---|---|---|---|
| ? | ✓ | ✓ | ✓ | ✓ |
| - | ✓ | - | - | ✓ |
| x | - | ✓ | ✓ | - |
| 0 | ✓ | - | - | - |
| u | - | ✓ | - | - |
| n | - | - | ✓ | - |
| 1 | - | - | - | ✓ |
| # | - | - | - | - |

| $(x, x')$ | $(0,0)$ | $(1,0)$ | $(0,1)$ | $(1,1)$ |
|---|---|---|---|---|
| 3 | ✓ | ✓ | - | - |
| 5 | ✓ | - | ✓ | - |
| 7 | ✓ | ✓ | ✓ | - |
| A | - | ✓ | - | ✓ |
| B | ✓ | ✓ | - | ✓ |
| C | - | - | ✓ | ✓ |
| D | ✓ | - | ✓ | ✓ |
| E | - | ✓ | ✓ | ✓ |

FIG. 5-1 – Notations utilisées dans [DCR06] pour représenter les différents états possibles pour un couple de bits $x$ et $x'$.

### 5-2.3 Différence généralisée

Nous devons cette approche à De Cannière et Rechberger, elle fût introduite pour la première fois dans leur article de 2006 [DCR06]. Il ne s'agit en fait pas à proprement parler d'un opérateur de différence, mais d'une représentation compacte du comportement des différences éventuelles. Cette représentation prend en compte et décrit tous les états possibles pour un couple de bits, et peut donc être utilisée pour décrire une caractéristique différentielle. Elle permet de plus, d'augmenter la quantité d'information disponible pour la cryptanalyse. La notation utilisée est celle de [DCR06] et est explicitée dans la figure 5-1.

La différentielle correspondant à un couple de mots $(X, Y)$ est notée $\nabla(X, Y)$. Nous reprenons ici l'exemple utilisé dans [Pey08], la différentielle pour un couple de mots $X$ et $Y$ de taille 8 bits

$$\nabla(X,Y) = \{(X,Y) | x_7 \wedge y_7 = 0, x_i = y_i \text{ pour } 2 \leq i \leq 5, x_1 \neq y_1, x_0 = y_0 = 0\}$$

est représentée par

$$\nabla(X,Y) = \texttt{[7?---x0]}.$$

L'utilisation de cette représentation a conduit à améliorer de façon significative les cryptanalyses de SHA-0 et SHA-1. Elle a permit de plus, de rendre plus lisibles les descriptions des caractéristiques différentielles. La figure 5-2 donne un exemple de représentation de la caractéristique linéaire correspondant à une collision locale.

## 5-3 Caractéristique linéaire

La caractéristique linéaire est l'élément clé des attaques par collisions contre les fonctions SHA-0 et SHA-1. Nous rappelons que cette caractéristique inclut à la fois la représentation des séquences de perturbations/corrections introduites au niveau du message et la représentation du chemin différentiel suivi par les différents états du registre $A$ lorsque le comportement de chaque collision locale est idéal. Dans la première étape de la cryptanalyse, la caractéristique linéaire est donc définie par la donnée d'un vecteur de perturbations (des conditions supplémentaires portant sur les bits du message expansé pourront être imposées ultérieurement). En effet on peut

| pas | $\nabla_{A_i}$ | $\nabla_{W_i}$ |
|---|---|---|
| | ... | ... |
| $i-1$ | `--------------------------------` | `--------------------------------` |
| $i$ | `--------------------------------` | `--------------------------------u-` |
| $i+1$ | `------------------------------u-` | `------------------------n-------` |
| $i+2$ | `--------------------------------` | `------------------------------n-` |
| $i+3$ | `--------------------------------` | `x-------------------------------` |
| $i+4$ | `--------------------------------` | `x-------------------------------` |
| $i+5$ | `--------------------------------` | `x-------------------------------` |
| $i+6$ | `--------------------------------` | `--------------------------------` |
| | ... | ... |

FIG. 5-2 – Caractéristique linéaire correspondant à une collision locale initiée au pas $i$ et à la position $j = 1$. La colonne de droite représente les différents états des mots de message expansés. Elle rend compte du fait que la collision insérée est de direction descendante selon la terminologie de [CJ98], de signe négatif selon la terminologie associée à la différence binaire signée. La colonne de gauche représente les différents états du registre $A$.

non seulement construire à partir de ce vecteur le masque de perturbation, mais aussi en déduire directement le chemin différentiel idéal du registre $A$.

Notre recherche de vecteurs de perturbations s'inscrit dans le cadre d'une cryptanalyse, l'objectif est donc de produire des vecteurs susceptibles de minimiser la complexité d'une attaque. D'où la nécessité de définir et d'évaluer l'efficacité de chaque vecteur candidat. L'expérience montre que cela n'est pas une tâche aisée car différentes méthodes ont été employées pour évaluer la complexité des cryptanalyses menées contre les fonctions SHA-0 et SHA-1. De plus, le choix d'un vecteur de perturbations doit aussi tenir compte des interactions possibles avec les autres outils mis en oeuvre lors de la cryptanalyse : générateur automatique de caractéristiques non-linéaires et autres techniques d'accélération de recherche.

Nous présentons ici les différentes contraintes imposées aux vecteurs de perturbations ainsi que différentes heuristiques de recherche utilisées pour les fonctions SHA-0 et SHA-1.

### 5-3.1 Contraintes et cas pathologiques

Le vecteur de perturbations représente l'ensemble des collisions locales appliquées aux mots de message expansé. L'objectif des cryptanalyses que nous considérons est de construire une collision pour la fonction de hachage. Afin d'atteindre cet objectif, on peut tenter d'obtenir une collision pour la fonction de compression (collision sur un bloc), on doit alors imposer au vecteur de perturbations de ne pas initier de collision locale lors des 5 derniers pas de la fonction de mise à jour des registres. Dans le cas des collisions sur plusieurs blocs (technique des blocs multiples), on peut relâcher cette contrainte.

Le vecteur de perturbations doit satisfaire la relation de récurrence de la fonction d'expansion de message, l'existence de collisions tronquées découlent de cette obligation. Dans les premières attaques conduites contre SHA-0 ces collisions tronquées

devaient êtres évitées, car leur prise en charge nécessite l'utilisation d'une caractéristique non-linéaire qui a été introduite postérieurement à ces attaques. Afin d'éviter les collisions tronquées, une contrainte supplémentaire est imposée au vecteur de perturbations.

La relation de récurrence créé un entrelacement des collisions locales. Cet entrelacement peut conduire dans certains cas à des comportements particuliers, par exemple un chemin différentiel impossible. Nous désignons ces cas particuliers par le terme de cas pathologiques. Un certain nombre de cas pathologiques ont été identifiés dans la littérature. Les premiers ont été exhibés par Chabaud et Joux dans leur article de 1998. Ils concernent les collisions locales consécutives se déroulant lors de la ronde $IF$ et de la ronde $MAJ$. En effet, deux collisions locales consécutives positionnées sur le même bit se déroulant dans une de ces deux rondes conduisent à un chemin différentiel impossible. Pour la fonction $IF$, cela a conduit les auteurs à imposer au vecteur de perturbations de ne pas présenter deux collisions locales consécutives lors des 16 premiers pas de la fonction de mise à jour des registres. Remarquons, que cette contrainte héritée des premières attaques contre la fonction SHA-0, est devenue caduque avec l'utilisation des caractéristiques non-linéaires. Pour la fonction $MAJ$, le problème est résolu en imposant des directions opposées aux perturbations consécutives apparaissant entre les pas 36 et 55. Nous avons mené une évaluation statistique des probabilités de bon comportement des collisions locales dans [Man10]. Cette étude a fait apparaître de nouveaux cas pathologiques jusqu'alors non répertoriés.

Un autre cas pathologique est l'effet de propagation de la retenue (*Carry Effect*) qui permet de "compresser" ou d'"étendre" des collisions locales adjacentes (démarrant au même pas sur des positions de bits consécutives). Cet effet peut faire l'objet d'un contrôle de l'attaquant. Il est d'ailleurs un des éléments principaux mis en oeuvre afin de construire la caractéristique non-linéaire utilisée dans [WYY05d].

### 5-3.2 Recherche de vecteurs de perturbations pour SHA-0

La fonction d'expansion définie pour la fonction SHA-0 n'entrelace pas les bits des mots de message expansés. Cette propriété simplifie grandement la recherche de bons vecteurs de perturbations. D'une part l'attaquant peut choisir de positionner les collisions locales sur la même position de bit, et d'autre part le nombre de collisions locales reste limité (certaines collisions étant "compensées" au fur et à mesure de la progression de l'expansion). Selon le tableau de probabilités théoriques de bon comportement, la position numéro 1 permet d'obtenir la meilleure probabilité de bon comportement. Il en découle que le nombre de bons candidats possibles est limité à $2^{16} - 1$ parmi lesquels on effectue une recherche exhaustive. L'évaluation d'un vecteur candidat repose essentiellement sur son poids de Hamming calculé à partir d'un certain pas qui diffère en fonction des techniques d'accélération de recherche (bits neutres, modifications de message ou boomerangs) mises en oeuvre dans la suite de l'attaque.

**Vecteur de perturbations de Chabaud et Joux.** La figure 5-3 présente le vecteur de perturbations proposé par Chabaud et Joux dans [CJ98]. On peut remarquer qu'il vérifie les contraintes de collision sur un seul bloc, sur les collisions tronquées ainsi que sur l'absence de collisions locales consécutives sur les 16 premiers pas. Ce

vecteur proposé en 1998 correspond à une approche entièrement linéaire de l'attaque. La complexité de l'attaque fondée sur ce vecteur et décrite dans l'article s'élève à $2^{61}$ évaluations de la fonction de compression.

| | | |
|---|---|---|
| 00000 | 00100010000000101111 | 01100011100000010100 |
| | 01000100100100111011 | 00110000111110000000 |

FIG. 5-3 – Vecteur de perturbations de Chabaud et Joux [CJ98].

**Vecteurs de perturbations de Biham *et al.*.** La figure 5-4 présente les quatre vecteurs de perturbations utilisés dans l'attaque de Biham *et al.* [BCJ05] qui a conduit à l'obtention de la première collision pour SHA-0. Ces vecteurs marquent le début de l'utilisation des propriétés non-linéaire de la fonction de ronde $IF$ et de la mise en oeuvre de la technique des blocs multiples. Les conditions portant sur les collisions tronquées ainsi que sur l'absence de collision locale sur les cinq derniers pas ne sont pas seulement relâchées mais mise à profit pour construire l'attaque. La complexité donnée dans l'article en nombre de paire de messages à traiter est égale à $2^{51}$, ce qui correspond à $2^{49}$ évaluations de la fonction de compression.

| | | |
|---|---|---|
| 00000 | 00010000101001000111 | 10010110000011100000 |
| | 00000011000000110110 | 00000110001011011000 |
| 01000 | 10000000010000101001 | 00011110010110000011 |
| | 10000000000011000000 | 11011000000110001011 |
| 10001 | 00100101000100101111 | 11000010000100001100 |
| | 00101100100000000001 | 11010011101000010001 |
| 11010 | 00100000000100001010 | 01000111100101100000 |
| | 11100000000000110000 | 00110110000001100010 |

FIG. 5-4 – Vecteurs de perturbations de Biham *et al.* [BCJ05].

**Vecteur de perturbations de Wang *et al.*** La figure 5-5 présente le vecteur de perturbations utilisé par Wang *et al.* dans [WYY05d]. L'utilisation d'une caractéristique non-linéaire a permis de relâcher les contraintes portant sur les collisions tronquées et la contrainte liée aux collisions locales consécutives dans la ronde de $IF$. La seule contrainte restante est celle liée à l'obtention d'une collision sur un seul bloc. La collision exhibée dans l'article est une collision sur deux blocs. Cependant, le premier bloc n'est employé que pour satisfaire des conditions sur un certain nombre de bits de la variable de chaînage liées à la caractéristique non-linéaire utili-

sée. La complexité de l'attaque associée est évaluée à $2^{39}$ évaluations de la fonction de compression.

| | | |
|---|---|---|
| 00111 | 01111001010010101000 | 01100010001011100000 |
| | 01000001000000000100 | 11001101011101000000 |

FIG. 5-5 – Vecteur de perturbations de Wang *et al.* [WYY05d].

**Vecteur de perturbations de Manuel et Peyrin.** La figure 5-6 présente le vecteur de perturbations utilisé par Manuel et Peyrin dans [MP08]. La combinaison de la caractéristique non-linéaire et de la technique des blocs multiples autorise un relâchement de toutes les contraintes. Cette attaque utilise de façon complète les propriétés non-linéaires. La complexité de l'attaque associée est mesurée à $2^{33}$ évaluations de la fonction de compression et reste à ce jour la meilleure attaque par collision contre la fonction SHA-0.

| | | |
|---|---|---|
| 11110 | 10001000110101101010 | 10000010000000101101 |
| | 00101000101001011001 | 11000010101010010100 |

FIG. 5-6 – Vecteur de perturbations de Manuel et Peyrin [MP08].

### 5-3.3 Vecteurs de perturbations pour SHA-1

La fonction d'expansion de SHA-1, à la différence de celle de SHA-0, entrelace les bits des mots de message expansés. Le résultat immédiat est qu'il n'est plus possible de confiner la position initiale des collisions locales à une seule position de bit. La permutation circulaire d'un bit vers la gauche limite le phénomène de compensation des collisions locales et assure une diffusion des positions de bit. La figure 5-7 illustre cette diffusion.

Cette diffusion modifie non seulement la taille de l'espace de recherche, mais aussi l'évaluation des vecteurs candidats. En effet, les positions de bit n'étant plus restreintes, les différentes positions de bit des collisions locales doivent à présent être prises en compte pour évaluer leur probabilité de bon comportement. Les solutions utilisées pour la recherche de bons vecteurs de perturbations consistent dans un premier temps à limiter la taille de l'espace de recherche puis à évaluer ces vecteurs en se fondant sur une heuristique particulière.

#### 5-3.3.1 Restriction de l'espace de recherche

L'approche la plus populaire se fonde sur l'algorithme décrit par Wang *et al.* dans [WYY05b] appelé *Rectangle Range*. Le sous-espace choisi pour rechercher les vecteurs de perturbations candidats est défini de la façon suivante :

$$\{W_i = (0, \ldots, 0, w_{i,1}, w_{i,0}) | i = t, \ldots, t+15\} \text{ pour } t = 0, \ldots, 64.$$

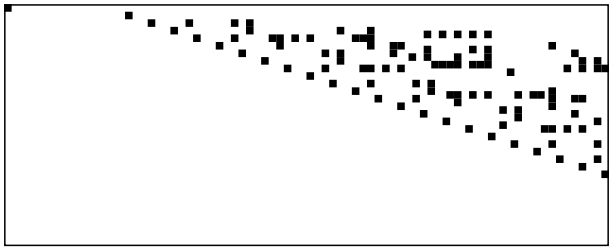


FIG. 5-7 – Illustration de la diffusion des positions des collisions locales pour la fonction SHA-1. Cette figure correspond à l'insertion d'une unique perturbation sur le premier bit du premier mot du message.

Cette approche consiste à définir une fenêtre au sein de laquelle les possibilités de positionnement des perturbations initiales sont limitées aux positions de bit 0 et 1. Cette restriction est directement issue du fait que la meilleure probabilité théorique de bon comportement pour une collision locale est attendue à la position 1. La taille de l'espace de recherche est alors égale à $65 \times 2^{32}$. Le même principe de restriction de l'espace de recherche est utilisée dans les travaux de Yajima *et al.* [YIN08], avec une extension à la position de bit 31. La taille de l'espace de recherche considéré étant alors égale à $65 \times 2^{48}$.

Nous avons proposé dans [Man09, Man10] un nouvel algorithme de recherche fondé sur l'utilisation d'une fenêtre d'information et d'un compromis différent. Ce compromis consiste à relâcher les contraintes imposées sur les positions de bit où sont insérées les collisions locales, pour la remplacer par une contrainte sur le nombre de collisions locales autorisées dans la fenêtre d'information. Cet algorithme détaillé au chapitre 7 section 7-2, fournit de bons résultats et est à l'origine de la classification des vecteurs de perturbations efficaces.

#### 5-3.3.2 Heuristiques d'évaluation

Les algorithmes de recherche utilisés par Wang *et al.* et Yajima *et al.* se fondent sur l'approche déterministe pour évaluer les vecteurs de perturbations candidats. Elle consiste à compter le nombre de conditions devant être respectées pour chacune des collisions locales présentes sur le chemin différentiel induit. Une collision locale initiée sur la position de bit 1 au sein d'une ronde de $XOR$ donne 2 conditions, 3 pour une position de bit 31 et 4 pour une position différente. Pour la ronde de $IF$ (respectivement $MAJ$) on a 5 (respectivement 4) conditions quelle que soit la position de bit considérée. Les collisions se déroulant à cheval sur deux fonctions de rondes sont évaluées au cas par cas. De plus, une technique dite de compression de bits (*Bit Compression*) est utilisée. Cette technique s'appuie sur l'effet de propagation de la retenue tel que décrit dans [WYY05d] pour "compresser" plusieurs collisions locales en une seule. En effet, un choix pertinent des directions affectées aux collisions locales permet en imposant des conditions supplémentaires au mots de message expansés d'obtenir sous certaines conditions une probabilité de bon comportement équivalente à la première collision locale considérée isolément. Les travaux de Yajima *et al.* présentés dans [YIN09] décrivent un certain nombre d'heuristiques additionnelles destinées à affiner l'évaluation des vecteurs candidats.

#### 5-3.3.3 Autres heuristiques

Certains auteurs ont utilisé des algorithmes liés à la théorie des codes pour la recherche de vecteurs de perturbations [JP05, PRR05]. Jutla et Patthak [JP05] ont calculé une probabilité théorique moyenne de bon comportement des collisions locales. Une fois cette probabilité moyenne définie, la recherche de bons vecteurs se résume à trouver des mots de petits poids de Hamming dans le code défini par l'expansion de message de SHA-1. L'algorithme de recherche de mots de petits poids employé dans ce papier se fonde sur l'algorithme de Léon [Leo88]. Nous avons en coopération avec Matthieu Finiasz et Mathieu Cluzeau effectué une recherche sur plusieurs semaines fondée sur une version optimisée de l'algorithme de Canteaut-Chabaud [CC98]. Cependant, l'utilisation de cet outil n'a pas permis de mettre à jour des vecteurs plus intéressants que ceux déjà obtenus au moyen de l'algorithme que nous avons proposé dans [Man09].

#### 5-3.3.4 Remarques

Contrairement à la façon dont nous avions procédé pour les vecteurs de perturbations pour SHA-0, nous ne présentons pas ici les différents vecteurs proposés pour les cryptanalyses. Dans le chapitre 7 consacré à l'amélioration de la caractéristique linéaire, nous présentons un travail de synthèse qui établit une classification des vecteurs de perturbations et fournit une étude comparative des différents vecteurs proposés dans la littérature. Ces travaux ont fait l'objet d'une présentation lors de la conférence WCC 2009 [Man09].

Nous remarquons de plus, que les méthodes d'évaluations publiées à ce jour prennent en compte de façon limitée les possibles interactions liés à l'enchevêtrement des collisions locales. Nous avons mené dans le cadre de cette thèse un certain nombre d'expérimentations sur ces interactions. Ces expérimentations ont mis à jour un certain nombre de comportements qui conduisent à réévaluer les heuristiques utilisées jusqu'ici pour la recherche de vecteurs de perturbations. Une partie de ces expérimentations est publiée dans [Man10] et présentée au chapitre 8.

Enfin une dernière remarque porte sur le fait que l'évaluation des vecteurs de perturbations peut ne pas constituer à elle seule une mesure suffisante pour mener à bien une cryptanalyse. En effet, les outils utilisés pour construire une cryptanalyse contre SHA-0 ou SHA-1 peuvent se comporter différemment pour des vecteurs possédant une même évaluation. Nous avons été confrontés à ce problème lors de la recherche de collision pour SHA-0 que nous avons réalisée avec Thomas Peyrin [MP08]. Lors de la construction de la caractéristique non-linéaire, des vecteurs équivalents en terme d'évaluation conduisaient à des caractéristiques non-linéaires de pertinences différentes.

### 5-3.4 Instantiation du vecteur de perturbation

Dans la section 5-3.1 nous avons décrit le cas pathologique de la fonction de ronde $MAJ$ nécessitant un traitement particulier relativement aux directions de collisions locales consécutives. De même, lors de l'évaluation du comportement d'une collision locale, que nous avons réalisée section 4-4.4 du chapitre précédent, nous avons remarqué que des contraintes portant seulement sur des bits des mots de message expansé devaient aussi être vérifiées pour permettre un bon comportement. Par conséquent,

un couple de message $(M, M')$ candidat pour une recherche de collision doit vérifier plusieurs niveaux de contraintes linéaires :

- une différence binaire relativement au masque de perturbations $M' = M \oplus \mathcal{P}$,
- une différence binaire signée pour spécifier la direction de certaines collisions locales afin d'éviter de possibles cas pathologiques,
- des équations liant certains bits des mots de message expansés pour permettre un bon comportement des collisions locales.

Ces contraintes apparaissent sur les mots de message expansés, mais l'expansion étant linéaire, on peut facilement se ramener à un système d'équation devant être vérifié par le couple $(M, M')$. L'impact de ces équations est plus particulièrement perceptible lors de l'utilisation d'une caractéristique non-linéaire qui doit expressément tenir compte de ces équations. Les techniques d'accélération de recherche qui utilisent des différences auxiliaires, comme les boomerangs, doivent aussi s'assurer d'être compatibles avec ces équations.

L'ensemble des cryptanalyses de SHA-0 et de SHA-1 publiées à ce jour impose uniformément la même direction pour l'ensemble des collisions locales, à l'exception des cas pathologiques identifiés. Cependant, les travaux menés lors de cette thèse conduisent à reconsidérer ce choix uniforme des directions. Nous reviendrons sur ce point au chapitre 8.

## 5-4 Caractéristique non-linéaire

La caractéristique non-linéaire associée à une cryptanalyse par collision contre les fonctions SHA-0 et SHA-1 permet de modéliser la propagation non-linéaire d'une différence à l'intérieur des composants non-linéaires de la fonction de mise à jour des registres. La construction d'une telle caractéristique est un procédé significativement plus complexe que la recherche de caractéristiques linéaires (vecteurs de perturbations). Cependant, nous avons vu qu'il découle des vecteurs de perturbations les plus intéressants des collisions tronquées et des cas pathologiques qui doivent être pris en charge. Les composants non-linéaires des fonctions SHA-0 et SHA-1 sont les fonctions Booléennes des rondes $IF$ et $MAJ$ ainsi que l'addition modulo $2^{32}$. Les 20 premiers pas de la fonction de mise à jour des registres cumulent deux de ces éléments non-linéaires. Pourtant, le cryptanaliste possède un contrôle direct sur les 16 premiers mots de message expansés et par conséquent sur les valeurs des registres internes jusqu'au pas 16. Le principe d'une caractéristique non-linéaire consiste donc à tirer partie de ce contrôle sur l'état interne afin de rendre possible l'utilisation des vecteurs de perturbations les plus prometteurs. Les premiers chemins différentiels non-linéaires pour les fonctions SHA-0 et SHA-1 ont été introduits par Wang *et al.* [WYY05d, WYY05b]. Les auteurs ont déclaré avoir construit ces chemins *à la main*, ce qui limitait l'application de cette technique à d'autres configurations de cryptanalyse. Dès lors, les recherches se sont principalement orientées vers la construction d'outils automatisés susceptibles de construire des caractéristiques non-linéaires à la demande [DCR06, DCM07, SKP07, YSN07, JP07, MP08].

Dans ce chapitre nous commencerons par décrire des éléments de la caractéristique non-linéaire mise en oeuvre par Wang *et al.* pour la cryptanalyse de SHA-0, puis nous décrirons les principes de l'outil de génération automatique de caractéristiques non-linéaires introduit pour la première fois par De Cannière et Rechberger.

### 5-4.1 Approche de Wang *et al.*

L'introduction de la caractéristique non-linéaire par Wang *et al.* a fondamentalement révolutionné la cryptanalyse des fonctions SHA-0 et SHA-1. Les chercheurs avaient constaté que les vecteurs de perturbations les plus prometteurs d'un point de vue de la complexité de l'attaque, possédaient des collisions tronquées, et/ou le cas pathologique de la fonction $IF$, et/ou conduisaient seulement à l'obtention de quasi-collisions. La contribution apportée par Wang *et al.* a été de fournir les moyens qui permettent d'utiliser des vecteurs de perturbations présentant des collisions tronquées et/ou des collisions locales consécutives dans la ronde de $IF$. Le premier moyen proposé réside dans l'approche déterministe et la différence binaire signée que nous avons décrites précédemment. Le second se présente sous la forme de la caractéristique non-linéaire.

Les caractéristiques données dans les deux articles [WYY05d, WYY05b] sont réputées avoir été construites à la main. De plus, bien que très complexes, les processus ayant conduit à l'obtention de ces caractéristiques ne sont pas détaillés. Seules quelques pistes et explications succinctes sont présentées dans les articles correspondants. Nous sommes parvenus lors d'un travail réalisé en 2006 [Man06] à reconstituer la caractéristique non-linéaire utilisée lors de la cryptanalyse de SHA-0, ce qui nous a amené à identifier et corriger certaines erreurs présentes dans l'article original. Nous ne présenterons cependant ici, que ce que nous considérons être les clés qui ont permis la création de cette caractéristique non-linéaire.

Nous allons tout d'abord introduire une notation supplémentaire. Pour indiquer la différence binaire signée $\Delta^{\pm}(W_i, W'_i) = +2^j$ nous utiliserons la notation $\overset{[+j]}{W_i}$. Par extension, une différence binaire signée $\Delta^{\pm}(A_i, A'_i) = +2^r - 2^s + 2^t$ sera notée $\overset{[+r,-s,+t]}{A_i}$. Afin d'améliorer la lisibilité, nous pourront être amenés à simplifier la notation de cette différence directement en $[+r, -s, +t]$ quand le contexte désigne clairement à qu'elle paire de mots elle s'applique. Nous nous référerons aux notations $+r$, $-s$ et $+t$ comme étant les valeurs de la différence $[+r, -s, +t]$.

**Effet de propagation de la retenue.** L'effet de propagation de la retenue où "*Carry effect*" se fonde sur l'utilisation d'une propriété particulière des puissances de 2. En effet, on a :

$$2^j = -2^j - 2^{j+1} \ldots - 2^{j+k-1} + 2^{j+k} \quad \forall (k, j) \in \mathbb{N}^2.$$

Cette propriété appliquée à un mot binaire, permet d'étendre une perturbation sur un bit d'indice $j$ à des bits d'indices supérieurs sans modifier la valeur arithmétique associée à ce mot. Soit $X$ un mot de 32 bits, posons :

$$X' = X + 2^4, \text{ et}$$
$$X'' = X - 2^4 - 2^5 - 2^6 + 2^7.$$

$X'$ et $X''$ forment deux mots de 32 bits différents, cependant quel que soit $Y$ un autre mot de 32 bits, on aura $X' + Y = X'' + Y$. Du point de vue de l'addition modulo $2^{32}$ la différence binaire signée $[+5]$ est équivalente à la différence binaire signée $[-4, -5, -6, +7]$. Nous pouvons de plus remarquer que l'égalité arithmétique est symétrique. 4 Il est donc possible d'utiliser cette propriété pour étendre une

différence, ou bien compresser plusieurs différences en une seule. C'est sur cette observation que repose le principe de la compression de bits.

Voici un autre exemple d'application de cette propriété. Soient $X$, $Y$ et $Z$ trois mots de 32 bits, et posons $W = X + Y + Z$. Soient $X'$, $Y'$ et $Z'$ définis de la façon suivante :

$$X' = X + 2^6,$$
$$Y' = Y + 2^6,$$
$$Z' = Z - 2^7,$$

On obtient alors :

$$X' + Y' + Z' = W = X + Y + Z.$$

Si nous considérons que cette équation reflète le même type d'équations vérifiées par la fonction de mise à jour des registres, nous pouvons formuler cette équation différemment. Les trois différentielles $\Delta^{\pm}(X, X') = [+6]$, $\Delta^{\pm}(Y, Y') = [+6]$ et $\Delta^{\pm}(Z, Z') = [-7]$ se sont compensées. On a donc corrigé une perturbation "descendante" sur la position de bit 6 de $Z$ par la combinaison de deux perturbations "montantes" sur la position de bit 7 de $X$ et de $Y$.

La combinaison de la différence binaire signée et de l'effet de propagation de la retenue permet de mieux visualiser la propagation des perturbations dans les registres, et offre aussi des possibilités supplémentaires (extension ou compression de différences) pour contrôler ces perturbations.

**Détournement de la fonction $IF$.** Paradoxalement, c'est la non-linéarité introduite par la fonction $IF$ qui est utilisée pour rendre la cryptanalyse possible. Nous avons remarqué section 5-2.2 de ce chapitre qu'en combinaison avec l'approche signée, il était possible de déterminer les conditions nécessaires pour que la fonction $IF$ puisse absorber, préserver ou inverser le signe d'une différence appliquée à son entrée. Afin d'illustrer cette propriété, nous allons considérer son application au pas numéro 3 de la fonction de mise à jour des registres correspondant à l'attaque décrite dans l'article de Wang *et al.* [WYY05d].

Conformément à la notation que nous avons définie, nous pouvons le représenter de la façon suivante :

$$\overset{[+2,-10,-12,-17,-18,+19]}{A_3} = \overset{[+10,-11,+12,+13,-14,-17,\ldots,-26,+27]}{(A_2 \lll 5)} + IF(\overset{[-2,-7,-8,+9,-32]}{A_1}, A_0 \lll 30, B_0 \lll 30) + (C_0 \lll 30) + \overset{[+2,+7,-32]}{W_2} + K_i$$

Ce qui revient du point de vue des différences seules et en appliquant l'effet de propagation de la retenue pour les différences $\Delta^{\pm}(A_3, A'_3)$ et $\Delta^{\pm}(A_2 \lll 5, A'_2 \lll 5)$ afin d'en compresser les différences à :

$$[+2, -10, -12, +17] = [-10, -12, +17] + IF([-2, -7, -8, +9, -32]) + [+2, +7, -32]$$

Cette équation ne constitue pas un réel formalisme mathématique. Elle permet cependant, d'illustrer de façon simple l'état des différences. Pour que cette équation

ait un sens d'un point de vue arithmétique, la différence en sortie de la fonction $IF$ doit valoir $[-7, \pm 32]$, ce qui correspond au comportement suivant :

1. absorber les valeurs $-2$, $-8$ et $+9$ de la différence $[-2, -7, -8, +9, -32]$,
2. préserver le signe de la valeur $-7$,
3. préserver la valeur $-32$, mais sans forcément en conserver le signe.

Le comportement 1 sera acquis dès lors que les conditions $b_{0,4} = a_{0,4}$, $b_{0,10} = a_{0,10}$ et $b_{0,10} = a_{0,11}$ portant sur les bits des valeurs initiales $B_0$ et $A_0$ seront vérifiées. Le comportement 2 nécessite quant à lui que les équations $a_{0,9} = 1$ et $b_{0,9} = 0$ soient vraies. Le comportement 3 impose la condition $b_{0,2} = \neg a_{0,2}$. Cet exemple illustre le fait que la fonction $IF$ permet, au prix de conditions supplémentaires, d'intervenir directement sur la propagation des différences au sein des registres.

Avec l'aide des outils que nous venons de présenter, Wang *et al.* sont parvenus à construire la caractéristique non-linéaire présentée table 5-1.

| | Conditions | | | |
|---|---|---|---|---|
| Registres | 31 - 24 | 23 - 16 | 15 - 8 | 7 - 0 |
| $A_0$ | -------- | 1-1100-1 | --1--1-1 | 10------ |
| $B_0$ | -------- | -------- | -----$aa$0 | ----$a$-$\bar{a}$- |
| $A_1$ | 1------- | 0$a$0011$a$0 | $aa$1-10$a$0 | 11----1- |
| $A_2$ | 0-0----- | --011111 | 111111-1 | 0010$a$--- |
| $A_3$ | 1-1--$aaa$ | $aa$0-0011 | 0010101- | -010100- |
| $A_4$ | 1----$a$-0 | 11111000 | --1111-0 | 110011-- |
| $A_5$ | 0------0 | -0001001 | 00100-0- | 01-01--- |
| $A_6$ | -------0 | -1011110 | 010-100- | -0--100- |
| $A_7$ | 0------1 | $a$1011111 | 0100--00 | 0----10- |
| $A_8$ | -------- | 10000000 | 00000-11 | 1---1--- |
| $A_9$ | 1------- | ---00000 | 0011001- | ----0--- |
| $A_{10}$ | -------- | ---11111 | 1111111- | ------0- |
| $A_{11}$ | 0------- | -------- | ------0- | ----1--- |
| $A_{12}$ | 0------- | -------- | -------- | 0---0--- |
| $A_{13}$ | -------- | -------- | -------- | 1---0-0- |
| $A_{14}$ | 1------- | -------- | -------- | ----1--- |
| $A_{15}$ | 0------- | -------- | -------- | ----1-1- |
| $A_{16}$ | 1------- | -------- | -------- | ----0--- |
| $A_{17}$ | 0------- | -------- | -------- | ------1- |
| $A_{18}$ | 1------- | -------- | -------- | -------- |
| $A_{19}$ | -------- | -------- | -------- | -------- |
| $A_{20}$ | -------- | -------- | -------- | -------- |

TAB. 5-1 – Conditions sur les bits de l'état interne correspondant à la caractéristique non-linéaire utilisée dans [WYY05d]. La notation $a$ (respectivement $\bar{a}$) indique que la valeur du bit correspondant est égale (respectivement opposée) à la valeur du même bit du registre immédiatement précédent.

### 5-4.2 Approche de De Cannière *et al.*

Les deux caractéristiques non-linéaires données par Wang *et al.* ne sont valables que pour les deux vecteurs de perturbations que les auteurs ont choisis. Or il existe d'autres vecteurs plus intéressants : les vecteurs conduisant à l'obtention d'une quasi-collision qui peuvent être utilisés en combinaison avec la technique des blocs multiples. En effet, bien que la collision exhibée pour SHA-0 utilise deux blocs de message, elle ne constitue pas une utilisation de la technique des blocs multiples, le premier bloc ne servant qu'à obtenir une variable de chaînage vérifiant un certain nombre de conditions imposées par la caractéristique non-linéaire. L'inconvénient majeur des techniques utilisées par Wang *et al.* réside dans le fait que leurs caractéristiques non-linéaires (pour SHA-0 et SHA-1) ont été obtenues à la main, ce qui rend fastidieuse une répétition du processus pour de nouveaux vecteurs de perturbations.

Différentes approches ont été menées afin d'automatiser la construction des caractéristiques non-linéaires, nous pouvons citer les travaux de Sugita *et al.* [SKP07] ou de Yajima *et al.* [YSN07]. Cependant, l'algorithme qui produit actuellement les meilleurs résultats repose sur les travaux de De Cannière et Rechberger [DCR06]. L'utilisation de générateurs automatiques de caractéristiques non-linéaires a conduit à l'obtention de collisions pour des versions réduites à 64 pas [DCR06], 70 pas [DCM07, JP07] et tout dernièrement à 72 et 73 pas [Gre10] de la fonction SHA-1.

Nous nous bornerons dans ce document à donner les principes qui gouvernent l'utilisation de ces générateurs de caractéristiques non-linéaires. En effet, d'une part le travail de recherche effectué pendant cette thèse s'est plutôt consacré à l'amélioration de la partie caractéristique linéaire (présentée au chapitre 7), et d'autre part une présentation complète de l'implémentation réalisée par Thomas Peyrin est disponible dans sa thèse [Pey08]. Le principe essentiel de l'outil de De Cannière et Rechberger consiste à appréhender les chemins différentiels candidats sous la forme d'une caractéristique différentielle généralisée afin de construire un arbre de recherche. Puis cet arbre est parcouru, en utilisant des outils de recherche en profondeur (*Backtrack search*), pour construire un chemin différentiel valide. L'effort de calcul nécessaire à l'obtention d'une collision est alors évalué en dénombrant le nombre moyen de noeuds à parcourir.

La première étape consiste donc à modéliser cette caractéristique. Nous avons abordé cet aspect section 5-2.3 en présentant la différence généralisée. Soient deux mots $X$ et $X'$, nous désignerons le couple $(X, X')$ par la notation $X^\star$. Un couple $X^\star$ se conforme à une caractéristique différentielle $\nabla X$ si la différence entre $X$ et $X'$ est l'une des différences possibles définies par $\nabla X$ ($X^\star \in \nabla X$). Un chemin différentiel pour la fonction de compression de SHA-0 ou de SHA-1, est donc constitué d'un ensemble de caractéristiques différentielles $\nabla A_{-4}, \nabla A_{-3}, \ldots, \nabla A_{80}$ pour les états des registres et d'un ensemble de caractéristiques différentielles $\nabla W_0, \ldots, \nabla W_{79}$ pour les mots de message expansés. Une paire de messages se conforme au chemin différentiel jusqu'au pas $i$ si :

$$\begin{cases} A^\star_{-4} \in \nabla A_{-4}, \ldots, A^\star_i \in \nabla A_i \\ W^\star_0 \in \nabla W_0, \ldots, W^\star_{i-1} \in \nabla W_{i-1} \end{cases} \tag{5-1}$$

Dès lors qu'un choix de vecteur de perturbations a été fait et qu'une caractéristique linéaire a été établie, il devient nécessaire de construire un chemin différentiel pour l'attaque. Cependant, cela n'est pas suffisant : il faut aussi s'assurer que ce

chemin différentiel conduit à une attaque effective. Cela nous amène à la seconde étape de développement de l'outil qui consiste à évaluer la probabilité de succès d'un chemin différentiel donné. Dans ce but, De Cannière et Rechberger ont introduit deux nouvelles notions désignées sous les termes de probabilité incontrôlée (*Uncontroled Probability*) et de probabilité contrôlée (*Controled Probability*) dont voici les définitions :

**Définition 5.4 (Probabilité incontrôlée)**
*La probabilité incontrôlée notée $P_u(i)$ d'une caractéristique différentielle à un pas $i$ est la probabilité que la sortie $A^\star_{i+1}$ du pas $i$ se conforme à la caractéristique différentielle étant donné que toutes les entrées se conforment à leur caractéristique différentielle respective. Soit :*

$$P_u(i) = P\left[A^\star_{i+1} \in \nabla A_{i+1} | A^\star_{i-j} \in \nabla A_{i-j} \text{ pour } 0 \leq j \leq 5 \text{ et } W^\star_i \in \nabla W_i\right]$$

**Définition 5.5 (Probabilité contrôlée)**
*La probabilité contrôlée notée $P_c(i)$ d'une caractéristique différentielle à un pas $i$ est la probabilité qu'il existe au moins une paire de messages $W^\star_i$ conforme telle que la sortie $A^\star_{i+1}$ du pas $i$ se conforme à la caractéristique différentielle étant donné que toutes les entrées se conforment à leur caractéristique différentielle respective. Soit :*

$$P_c(i) = P\left[\exists W^\star_i \in \nabla W_i : A^\star_{i+1} \in \nabla A_{i+1} | A^\star_{i-j} \in \nabla A_{i-j} \text{ pour } 0 \leq j \leq 5\right]$$

Ces définitions permettent d'évaluer la probabilité qu'un chemin différentiel soit valide.

Afin de pouvoir estimer l'effort en terme de calcul, une nouvelle définition est nécessaire pour mesurer le nombre de degrés de liberté laissés au message.

**Définition 5.6 (Degrés de liberté)**
*Le degré de liberté $F(i)$ associé à une caractéristique différentielle à un pas $i$ est égal au nombre de façons de choisir $W^\star_i \in \nabla W_i$ en ne violant aucune des conditions imposées sur les mots de message expansés, étant données les valeurs fixées de $W^\star_j$ pour $0 \leq j < i$.*

À partir de ces définitions, nous pouvons exprimer le nombre $N(i)$ de noeuds visités à chaque pas de la fonction de mise à jour des registres lors de la recherche d'une collision. Étant donné que le nombre moyen de fils d'un noeud pour un pas $i$ est égal à $F(i).P_u(i)$, que seule une fraction $P_c(i)$ de ces noeuds possède au moins un fils et que la recherche s'arrête lorsque le pas $n$ est atteint, nous pouvons déduire la relation de récurrence suivante :

$$N(i) = \begin{cases} 1 \text{ si } i = n, \\ max\left\{\dfrac{N(i+1)}{(F(i).P_u(i)}, \dfrac{1}{P_c(i)}\right\} \text{ si } i < n, \end{cases} \tag{5-2}$$

où $n$ désigne le pas auquel on souhaite obtenir une collision : $n < 80$ pour une version réduite, $n = 80$ pour la version normalisée de SHA-0 ou SHA-1. Le nombre total de noeuds parcourus est alors égal à :

$$N(n) = \sum_{i=0}^{n} N(i).$$

Afin d'illustrer l'utilisation de ces principes, nous présentons dans les tableaux 5-2 et 5-3 le chemin différentiel correspondant au premier bloc de l'attaque que nous avons menée avec Thomas Peyrin contre la fonction SHA-0 [MP08]. Nous donnons pour chaque pas le nombre de degrés de liberté $F(i)$, le logarithme en base 2 de la valeur de la probabilité contrôlée $P_c(i)$ et du nombre moyen de noeuds visités $N(i)$.

| $i$ | $\nabla_{A_i}$ | $\nabla_{W_i}$ | $F(i)$ | $P_c(i)$ | $N(i)$ |
|---|---|---|---|---|---|
| -4 : | 00001111010010111000011111000011 | | | | |
| -3 : | 01000000110010010101000111011000 | | | | |
| -2 : | 01100010111010110111001111111010 | | | | |
| -1 : | 11101111110011011010101110001001 | | | | |
| 00 : | 01100111010001010010001100000001 | 01001110010010110000010l0n0010u1 | 0 | 0.00 | -10.00 |
| 01 : | 11101101111111111100111011n1111u0 | 01000000110110110101000001n000000 | 0 | 0.00 | -10.00 |
| 02 : | 01100111011111111101n10101101010 | 11110110000111110100111011n011011 | 0 | 0.00 | -10.00 |
| 03 : | 001101010010100n00001100n0uuuuu0 | 00111000101010010111001000000101 | 0 | 0.00 | -10.00 |
| 04 : | 111100000nu000001010110001110000 | u1100100010000000010100000u0101n1 | 0 | 0.00 | -10.00 |
| 05 : | 00111n000000100110001000000u0n1u1 | n0010100110010000101010010100000 | 0 | 0.00 | -10.00 |
| 06 : | 10110101110110110000101u100u1001 | n0100010111101000011111000u000100 | 0 | 0.00 | -10.00 |
| 07 : | 100unnnnnnnnn0100nu0100101u11001 | 00010010111101000101011001011010 | 0 | 0.00 | -10.00 |
| 08 : | 1000011100001n000n100u0n010nn001 | 01011010011100010001100011n0001u1 | 0 | 0.00 | -10.00 |
| 09 : | 0010000000000010un00nu1u1un01100 | n1010001010001111001011001u1110n0 | 0 | 0.00 | -10.00 |
| 10 : | 11100110100101000nu01u10un00n100 | 01111010011000100100011100011000 | 0 | 0.00 | -10.00 |
| 11 : | 01111000111000110lnuu10101000101 | 01110001001101110101100l1u1110u0 | 0 | 0.00 | -10.00 |
| 12 : | 01001101011010000010u0000n110000 | 10110111110101-----1-----u000001 | 10 | -4.00 | -10.00 |
| 13 : | 010110011100000----010-0-01001u0 | 101001010----------n0000u1 | 16 | -1.00 | -4.00 |
| 14 : | 10111100--------------1--110u011 | 01101-0----0--1----0---0-1-011u0 | 16 | -2.00 | 11.00 |
| 15 : | 10100------------------0-1-u0100 | n0101-0----0--1----0---0-1-11000 | 16 | -2.00 | 25.00 |
| 16 : | --01----------------------n0011 | 010001110----------00101n0 | 0 | 0.00 | 39.00 |
| 17 : | ----------------------------1n- | n1000-0----1--1----1---0-u-10011 | 0 | -2.00 | 39.00 |
| 18 : | 1---------------------------0-- | 01000-0----1--1----0---0-0-011u0 | 0 | 0.00 | 37.00 |
| 19 : | ------------------------------ | n00110100----------0001011 | 0 | 0.00 | 37.00 |
| 20 : | ------------------------------ | n0110-0----1-------0-----0-000u1 | 0 | -1.00 | 37.00 |
| 21 : | ----------------------------n- | u1100-1----------u-10111 | 0 | 0.00 | 36.00 |
| 22 : | ------------------------------ | 00001-1----------0-00110 | 0 | -2.00 | 36.00 |
| 23 : | ----------------------------n- | n1011-1----0-------0-----u-11001 | 0 | 0.00 | 34.00 |
| 24 : | ------------------------------ | u0000-0----------1-11100 | 0 | -2.00 | 34.00 |
| 25 : | ----------------------------n- | 01101-1----------u-10111 | 0 | 0.00 | 32.00 |
| 26 : | ------------------------------ | u1010-1----0-------1-----0-011u0 | 0 | -1.00 | 32.00 |
| 27 : | ------------------------------ | 01001-1----------0-01110 | 0 | 0.00 | 31.00 |
| 28 : | ------------------------------ | u0000-0----------1-11011 | 0 | 0.00 | 31.00 |
| 29 : | ------------------------------ | u0111-0----------0-00010 | 0 | 0.00 | 31.00 |
| 30 : | ------------------------------ | 01101-1----------1-10010 | 0 | 0.00 | 31.00 |
| 31 : | ------------------------------ | 10110-1----------0-01001 | 0 | 0.00 | 31.00 |
| 32 : | ------------------------------ | 00111-1----------1-00100 | 0 | 0.00 | 31.00 |
| 33 : | ------------------------------ | 01011-1----------1-11101 | 0 | 0.00 | 31.00 |
| 34 : | ------------------------------ | 00010-0----------0-010u0 | 0 | -1.00 | 31.00 |
| 35 : | ----------------------------u- | 10001-0----------n-10110 | 0 | 0.00 | 30.00 |
| 36 : | ------------------------------ | 11100-0----------0-000u1 | 0 | -1.00 | 30.00 |
| 37 : | ------------------------------ | n0010-0----------0-001u0 | 0 | -1.00 | 29.00 |
| 38 : | ----------------------------u- | n1101-0----------n-11110 | 0 | 0.00 | 28.00 |
| 39 : | ------------------------------ | n1100-1----------0-001n0 | 0 | -1.00 | 28.00 |
| 40 : | ------------------------------ | n1111-0----------0-10000 | 0 | -1.00 | 27.00 |
| | ... | ... | ... | ... | ... |

TAB. 5-2 – Exemple de chemin différentiel obtenu à l'aide d'un outil automatisé de génération de caractéristique non-linéaire. Le contenu du tableau correspond à la première partie (pas 1 à 40) du chemin différentiel du premier bloc utilisé par l'attaque publié dans [MP08].

| $i$ | $\nabla_{A_i}$ | $\nabla_{W_i}$ | $F(i)$ | $P_c(i)$ | $N(i)$ |
|---|---|---|---|---|---|
| | ... | ... | ... | ... | ... |
| 41 : | ---------- | n0010-1----------0-11010 | 0 | -1.00 | 26.00 |
| 42 : | ---------- | n0100-0----------1-110u1 | 0 | -1.00 | 25.00 |
| 43 : | ----------u- | 00000-1----------n-01010 | 0 | 0.00 | 24.00 |
| 44 : | ---------- | 00011-0----------0-100n0 | 0 | -1.00 | 24.00 |
| 45 : | ---------- | n0111-1----------1-10110 | 0 | -1.00 | 23.00 |
| 46 : | ---------- | n0111-1----------0-00010 | 0 | -1.00 | 22.00 |
| 47 : | ---------- | u0010-1----------1-00000 | 0 | 0.00 | 21.00 |
| 48 : | ---------- | 01101-0----------0-010n0 | 0 | -1.00 | 21.00 |
| 49 : | ----------n- | 11111-1----------u-10011 | 0 | 0.00 | 20.00 |
| 50 : | ---------- | 01000-1----------0-100u0 | 0 | -1.00 | 20.00 |
| 51 : | ---------- | u1110-1----------0-10010 | 0 | -1.00 | 19.00 |
| 52 : | ---------- | n1101-1----------1-11110 | 0 | -1.00 | 18.00 |
| 53 : | ---------- | n0001-1----------1-001u0 | 0 | -1.00 | 17.00 |
| 54 : | ----------u- | 11011-0----------n-11110 | 0 | 0.00 | 16.00 |
| 55 : | ---------- | 10001-0----------0-000n0 | 0 | -1.00 | 16.00 |
| 56 : | ---------- | n0111-1----------0-001n1 | 0 | -2.00 | 15.00 |
| 57 : | ----------n- | n0110-1----------u-11101 | 0 | -1.00 | 13.00 |
| 58 : | ---------- | u1110-1----------1-11001 | 0 | -2.00 | 12.00 |
| 59 : | ----------n- | u1110-0----------u-010u1 | 0 | -2.00 | 10.00 |
| 60 : | ----------u- | n1111-1----------n-100n1 | 0 | -1.00 | 8.00 |
| 61 : | ---------- | 01010-0----------0-010n1 | 0 | -1.00 | 7.00 |
| 62 : | ---------- | 01111-1----------1-11111 | 0 | 0.00 | 6.00 |
| 63 : | ---------- | 10011-1----------0-00010 | 0 | 0.00 | 6.00 |
| 64 : | ---------- | n1000-0----------0-10110 | 0 | 0.00 | 6.00 |
| 65 : | ---------- | 01000-0----------1-00011 | 0 | 0.00 | 6.00 |
| 66 : | ---------- | 01000-0----------0-101u1 | 0 | -1.00 | 6.00 |
| 67 : | ----------u- | 01001-0----------n-01001 | 0 | 0.00 | 5.00 |
| 68 : | ---------- | 10001-0----------0-100u0 | 0 | -1.00 | 5.00 |
| 69 : | ---------- | u0010-1----------1-11000 | 0 | 0.00 | 4.00 |
| 70 : | ---------- | u1010-0----------1-011n1 | 0 | -1.00 | 4.00 |
| 71 : | ----------n- | u0101-0----------u-01101 | 0 | 0.00 | 3.00 |
| 72 : | ---------- | 00011-1----------0-100u0 | 0 | -1.00 | 3.00 |
| 73 : | ---------- | n1010-1----------0-11000 | 0 | 0.00 | 2.00 |
| 74 : | ---------- | n1100-0----------0-10010 | 0 | 0.00 | 2.00 |
| 75 : | ---------- | u1110-1----------1-110n1 | 0 | -1.00 | 2.00 |
| 76 : | ----------n- | 11011-1----------u-00100 | 0 | 0.00 | 1.00 |
| 77 : | ---------- | 00111-0----------1-000n1 | 0 | -1.00 | 1.00 |
| 78 : | ---------- | n0011-0----------1-11101 | 0 | 0.00 | 0.00 |
| 79 : | ---------- | u0101-0----------1-01000 | 0 | 0.00 | 0.00 |
| 80 : | ---------- | | | | |

TAB. 5-3 – Exemple de chemin différentiel obtenu à l'aide d'un outil automatisé de génération de caractéristique non-linéaire. Le contenu du tableau correspond à la seconde partie (pas 41 à 80) du chemin différentiel du premier bloc utilisé par l'attaque publié dans [MP08].

## 5-5 Technique des blocs multiples

Le principe de construction des fonctions SHA-0 et SHA-1 repose sur l'utilisation d'un schéma de chiffrement par bloc ad hoc. Cependant, les schémas de chiffrement doivent posséder un caractère inversible afin de rendre possible le déchiffrement. Mais, cette propriété étant indésirable dans le contexte des fonctions de hachage, l'ajout d'une étape de rebouclage (*Feed Forward*) est nécessaire et conduit à obtenir la

construction dite de Davies-Meyer. Pour les fonctions SHA-0 et SHA-1, le rebouclage utilisé consiste à effectuer une somme modulo $2^{32}$ de l'état initial des 5 registres avec leur état final $(A_0 + A_{80}, B_0 + B_{80}, C_0 + C_{80}, D_0 + D_{80}, E_0 + E_{80})$. Paradoxalement, bien que conçu pour améliorer la résistance au calcul d'antécédent, le rebouclage des fonctions SHA-0 et SHA-1 est mis à profit par les cryptanalystes lors des attaques par collisions. Les différentes applications de la technique des blocs multiples à ces fonctions illustrent parfaitement la faiblesse de ce rebouclage.

Nous introduirons tout d'abord le principe général de la technique des blocs multiples avant de détailler le premier exemple d'application de cette technique à la fonction SHA-0 dû à Biham *et al.* [BCJ05]. Nous décrirons ensuite comment une utilisation appropriée du rebouclage en combinaison avec l'utilisation d'une caractéristique non-linéaire constitue aujourd'hui l'approche classique pour les attaques par collisions contre SHA-0 et SHA-1. Finalement, nous formulerons une remarque sur le cas des cryptanalyses proposées par Wang *et al.* [WYY05b, WYY05d].

### 5-5.1 Principe de la technique des blocs multiples.

La technique des blocs multiples a été utilisée avec succès dans un certain nombre des attaques publiées récemment. Cette technique ne s'applique qu'aux fonctions de hachage itératives et réside essentiellement dans la recherche de blocs de messages particuliers passés en arguments à la fonction de compression. Pour la décrire, nous allons tout d'abord introduire deux nouvelles définitions.

**Définition 5.7 (Quasi-collision)** *(Near Collision)*
*Soit une fonction de hachage cryptographique $h : \{0,1\}^* \to \{0,1\}^n$, munie d'une valeur initiale $IV$, on notera alors la fonction $h$ sous la forme $h(IV, .)$. Un couple de messages $(M, M') \in \{0,1\}^* \times \{0,1\}^*$, avec $M \neq M'$, forme une quasi-collision pour $h$ si*

$$h(IV, M) \approx h(IV, M').$$

On peut aussi formuler cette propriété de la façon suivante :

$$w_H(h(M) \oplus h(M')) \ll n,$$

où $w_H$ désigne le poids de Hamming et où la notation $\ll$ indique que le nombre de différences est petit devant la taille de l'empreinte.

**Définition 5.8 (Pseudo-collision)**
*Soit une fonction de hachage cryptographique itérative $h : \{0,1\}^* \to \{0,1\}^n$. Un couple de messages $(M, M') \in \{0,1\}^* \times \{0,1\}^*$, avec $M \neq M'$, et un couple de valeurs initiales $(V, V')$, forment une pseudo-collision pour $h$ si*

$$h(V, M) = h(V', M')$$

Cette définition englobe les cas où $V = V'$ et $V \neq V'$ ; si $V = V' = IV$, on retrouve la définition classique d'une collision. On pourra rencontrer dans la littérature des terminologies différentes : le cas $V = V'$ est parfois désigné sous le terme de collision libre et le cas $V \neq V'$ sous le terme de pseudo-collision. Une illustration de quasi-collision et pseudo-collision est proposée figure 5-8.

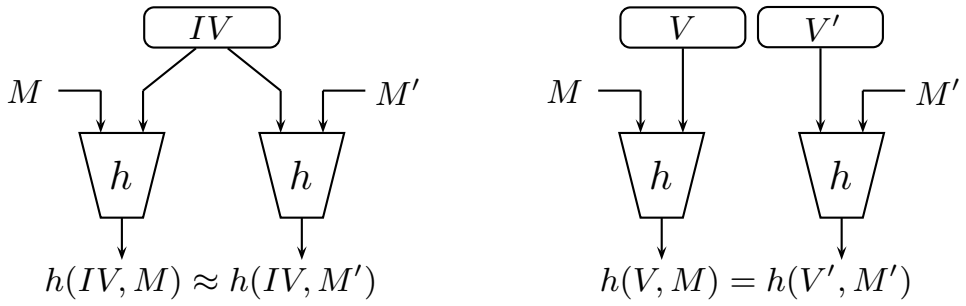



FIG. 5-8 – Quasi-collision et pseudo-collision.

Les définitions de quasi-collision et de pseudo-collision s'étendent naturellement aux fonctions de compression. La technique des blocs multiples, appliquée par exemple sur deux blocs de message, va consister dans une première étape à rechercher une quasi-collision pour la première itération de la fonction de compression $f$ :

$$(M_1, M_1') \quad \text{tel que} \quad f(IV, M_1) \approx f(IV, M_1').$$

Puis dans la seconde étape, à rechercher une pseudo-collision particulière correspondant à la seconde itération de la fonction de compression :

$$\begin{array}{l}(M_2, M_2') \quad \text{tel que} \quad f(H, M_2) = f(H', M_2'), \\ \text{où} \quad H = f(IV, M_1) \quad \text{et} \quad H' = f(IV, M_1').\end{array}$$

Une fois la quasi-collision et la pseudo-collision obtenues pour la fonction de compression, il suffit de concaténer les deux blocs pour construire deux messages qui forment une collision pour la fonction de hachage :

$$h_{IV}(M_1||M_2) = h_{IV}(M_1'||M_2')$$

Cette technique, illustrée figure 5-9, fait à présent partie intégrante de l'éventail des outils à la disposition des cryptanalystes.

### 5-5.2 Attaque de Biham *et al.*

La première collision obtenue pour la fonction SHA-0 est le fruit de la collaboration de plusieurs équipes de chercheurs et de la mise en oeuvre de la technique des blocs multiples. Le principe de cette attaque repose essentiellement sur trois observations :

- la caractéristique linéaire utilisée dans l'attaque est sous le contrôle total de l'attaquant qui peut définir le signe (respectivement à l'addition modulo $2^{32}$) des différences appliquées sur la variable de chaînage d'entrée et obtenues sur la variable de chaînage de sortie ;
- grâce au rebouclage, une différence en sortie peut être annulée par une différence en entrée, si leurs valeurs arithmétiques modulo $2^{32}$ sont opposées, ce qui donne un certain degré de liberté car deux mots binaires de 32 bits différents peuvent posséder une même valeur arithmétique (confert *Carry Effect*) ;

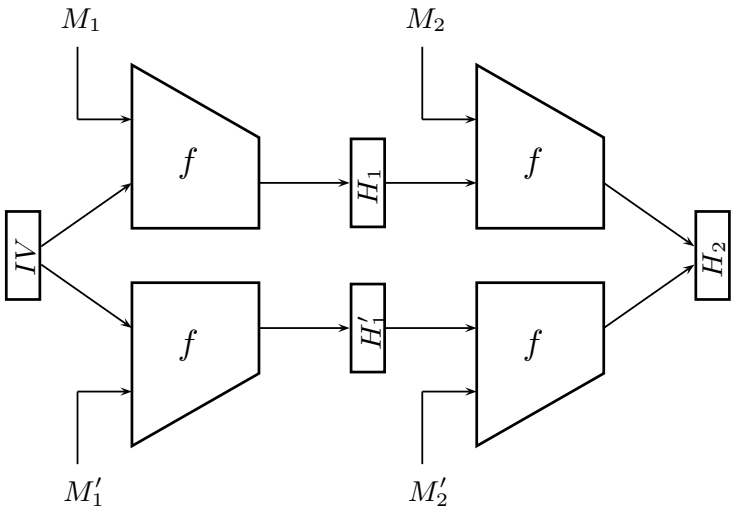



FIG. 5-9 – Technique des blocs multiples : collision sur 2 blocs.

- les 5 premières étapes de la fonction de ronde $IF$ (au sein desquelles s'expriment les différences sur la variable de chaînage d'entrée) peuvent induire, de part le caractère non-linéaire de cette fonction, un comportement identique pour deux différences d'entrée différentes.

Forts de ces observations, les auteurs de cette attaque ont construit un graphe orienté dont les noeuds représentent les différences présentes dans les variables de chaînage et les arêtes les caractéristiques linéaires qui joignent les différences d'entrée aux différences de sortie en prenant en compte le rebouclage. La recherche d'un chemin efficace dans ce graphe à conduit à l'obtention d'une collision constituée de quatre blocs de message et fondée sur quatre caractéristiques linéaires différentes. La figure 5-10 illustre la technique des blocs multiples telle qu'utilisée dans l'article de Biham *et al.*

La technique des blocs multiples mise en oeuvre par Biham *et al.*, n'est pas transposable en l'état à la fonction SHA-1. En effet, afin de construire un chemin différentiel passant par plusieurs blocs, il est nécessaire de construire un graphe connexe. Pour la fonction SHA-0, les perturbations sont contenues sur la position de bit 1, ce qui induit un nombre limité de sommets (nous avons 5 registres et donc $2^5 = 32$ sommets différents). Pour la fonction SHA-1, la permutation circulaire ajoutée à la fonction d'expansion étale les positions de bits où se situent les perturbations, augmentant exponentiellement le nombre de sommets, et rendant beaucoup plus difficile l'obtention d'un graphe suffisamment connexe et conduisant à une attaque avec une bonne complexité.

**Remarque.** Le principe de la technique des blocs multiples n'a été mis en oeuvre pour la cryptanalyse de SHA-0 et SHA-1 qu'à partir du moment ou les caractéristiques non-linéaire de la fonction $IF$ ont pu être exploitées. En effet, c'est l'utilisation adéquate de la non-linéarité qui autorise l'emploi de vecteurs de perturbations plus performants mais aboutissant seulement à des quasi-collisions. Dans le cadre d'une attaque linéaire, la technique des blocs multiples ne permet pas d'améliorer la com-

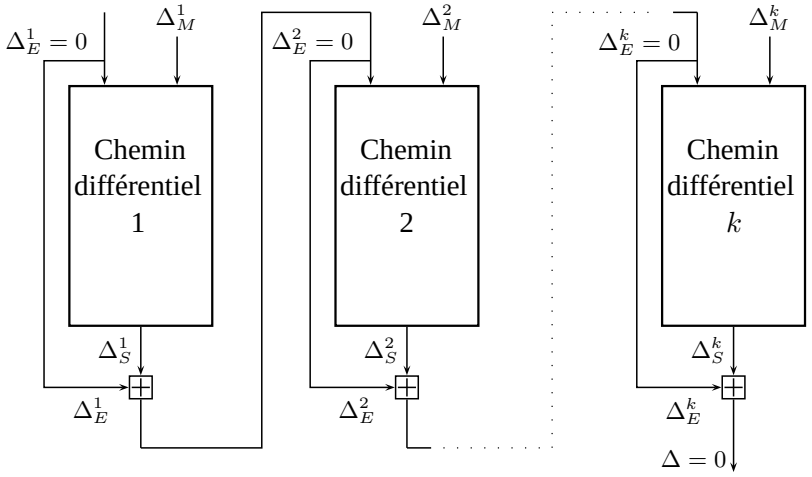



FIG. 5-10 – Principe de mise en oeuvre de la technique des blocs multiples utilisée par Biham *et al.* dans [BCJ05].

plexité de l'attaque de façon significative.

### 5-5.3 Forme actuelle

Nous avons constaté dans la section consacrée à la caractéristique non-linéaire que celle-ci permet de s'affranchir des conditions correspondant aux collisions tronquées et au cas pathologique de la fonction $IF$, ce qui autorise l'utilisation de vecteurs de perturbations plus performants. L'impact de la caractéristique non-linéaire est tout aussi intéressant relativement à la mise en oeuvre de la technique des blocs multiples. En effet il devient théoriquement possible, par le biais d'une caractéristique non-linéaire appropriée, de connecter une différence d'entrée quelconque à une caractéristique linéaire conduisant à une différence de sortie choisie.

La technique des blocs multiples utilisée par Biham *et al.* a nécessité l'utilisation de quatre caractéristiques linéaires connectées. Grâce à l'outil que constitue la caractéristique non-linéaire, une unique caractéristique linéaire peut être employée (conjointement à deux caractéristiques non-linéaires appropriées) pour conduire à une collision sur deux blocs de message. La figure 5-11 illustre la technique des blocs multiples sous la forme mise en pratique dans les dernières cryptanalyses de SHA-0 et SHA-1.

L'utilisation de la caractéristique non-linéaire permet de s'affranchir des problèmes de recherche de graphe connexe évoqués précédemment. Il en résulte que la combinaison caractéristique non-linéaire/technique des blocs multiples est très efficace pour les cryptanalyses des fonctions SHA-0 et SHA-1. La première application de la technique des blocs multiples sous cette forme est due à De Cannière et Rechberger [DCR06].

### 5-5.4 Attaque de Wang *et al.*

La paternité de l'utilisation de la technique des blocs multiples aux fonctions SHA-0 et SHA-1 est quelque fois attribuée à tort à Wang *et al.* Dans les faits, aucun

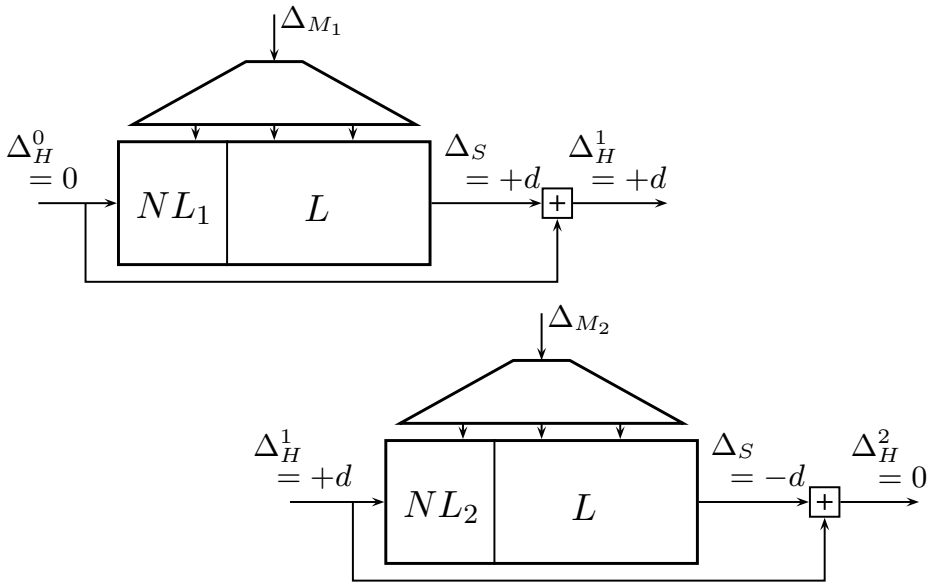



FIG. 5-11 – Technique des blocs multiples utilisée en pratique pour la recherche de collisions pour les fonctions SHA-0 et SHA-1.

des deux articles [WYY05b, WYY05d] ne la met en pratique.

Dans l'article consacré à la cryptanlyse de SHA-0 [WYY05d], la collision exhibée est effectivement obtenue au moyen de deux blocs de message. Cependant comme nous l'avons déjà mentionné, le premier bloc de message ne sert qu'à satisfaire 16 conditions héritées de la caractéristique non-linéaire. Ce premier bloc de message ne conduit pas à une quasi-collision et le vecteur de perturbations employé vérifie la condition imposant l'absence de début de collision locale sur les cinq derniers pas. Pour utiliser un vocabulaire technique, ce que les auteurs ont obtenu constitue une collision à début libre (*Free Start Collision*). Le fait qu'il n'y ait que seulement 16 conditions à remplir pour la variable de chaînage d'entrée permet de transformer cette collision à début libre en collision classique pour un coût négligeable. La figure 5-12 illustre la forme (en terme de blocs de message et de différences) de la collision pour SHA-0 présentée dans l'article.

Dans l'article concernant la cryptanalyse de SHA-1 [WYY05b], les auteurs présentent un chemin différentiel pouvant conduire à l'obtention d'une quasi-collision. Ils indiquent de plus, qu'il est possible de transformer cette quasi-collision en collision au moyen de la technique des blocs multiples. Cependant, la mise en oeuvre de cette technique nécessite l'utilisation de deux caractéristiques non-linéaires spécifiques dans le cas d'une attaque sur deux blocs. Les informations présentées dans l'article ne permettent pas de construire une telle attaque.

## 5-6 Techniques d'accélération de recherche de messages

La composante d'accélération de recherche des attaques par collision menées contre SHA-0 et SHA-1 constitue une partie essentielle de ces attaques. L'amélioration de la complexité de l'attaque obtenue grâce à ces techniques provient du fait

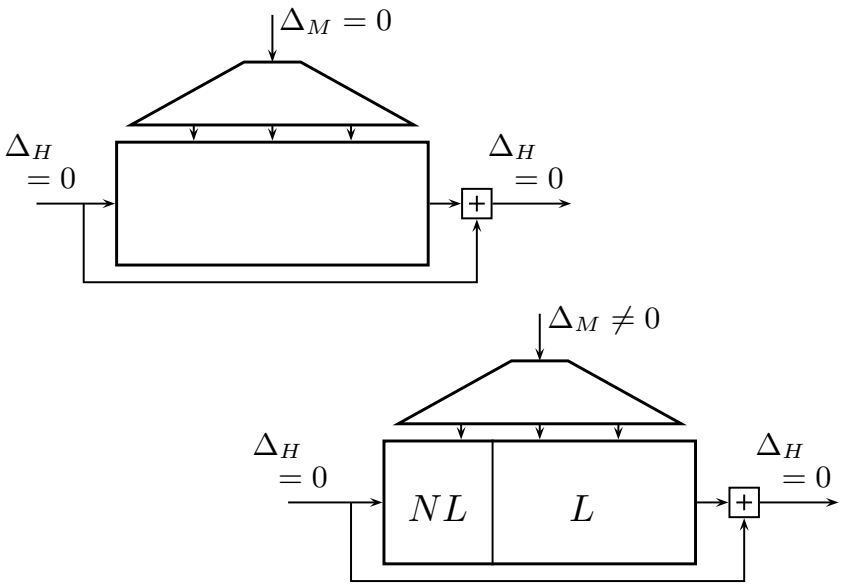



FIG. 5-12 – Illustration du point de vue des différences de la collision pour SHA-0 présentée dans [WYY05d].

que l'on peut considérer que le calcul du coût du chemin différentiel débute à un pas ultérieur au pas 16. En effet, ces techniques permettent de générer de façon efficace des paires de messages se conformant, avec une probabilité proche de 1, au chemin différentiel pour des pas supérieur à 16.

La première méthode d'accélération de recherche utilisée fut proposée par Biham et Chen en 2004 [BC04] sous la forme des bits neutres. Puis en 2005, Wang *et al.* [WYY05b] employèrent une technique dite de modification de message que les auteurs appliquèrent avec succès à plusieurs autres fonctions de hachage. En 2007, Joux et Peyrin [JP07] adaptèrent l'attaque boomerang de Wagner [Wag99], initialement destinée à la cryptanalyse des schémas de chiffrement par bloc, aux fonctions de hachage. Nous présentons dans ce chapitre chacune de ces trois méthodes.

### 5-6.1 Bits neutres

Biham et Chen ont mis en évidence une propriété particulière de la fonction de compression de SHA-0 : l'existence de bits neutres. Dans [BC04], les bits neutres sont utilisés pour trouver des quasi-collisions pour SHA-0 ainsi que des collisions pour des versions réduites.

**Principe des bits neutres.** Le principe de l'accélérateur de recherche proposé par Biham et Chen se fonde sur la notion de conformité qui est définie dans leur article de la façon suivante :

**Définition 5.9 (Conformité)**
*Étant donné un vecteur $\alpha = (\alpha_1, \ldots, \alpha_r)$, on dira qu'un couple de messages $(M, M')$ se conforme à $\alpha_r$ si les états des registres $A$ et $A'$ correspondants à $M$ et $M'$ vérifient :*

$$A_i \oplus A'_i = \alpha_i \quad \textit{pour} \quad i = 1, \ldots, r.$$

Le vecteur $\alpha$ représente l'évolution des différences des états des registres $A$ et $A'$, dans l'hypothèse où toutes les perturbations sont corrigées convenablement. Ce vecteur constitue une représentation de la caractéristique linéaire du point de vue du registre $A$. Si un couple $(M, M')$ se conforme à $\alpha_r$, cela implique qu'au pas $r$ les états de tous les registres se conforment exactement à la séquence de collisions locales définie par le vecteur de perturbations. Pour un couple de messages tiré aléatoirement dans l'ensemble des couples de messages se conformant à $\alpha_r$, les probabilités de bon comportement des collisions locales pour les pas inférieurs ou égaux à $r$ sont égales à 1. En conséquence, rechercher une collision à partir d'un ensemble de couples de messages qui se conforment à $\alpha_r$, revient dans les faits à commencer la recherche au pas $r$. En fonction de la valeur de $r$, cela peut réduire de façon significative la complexité de l'attaque.

Une fois définie la propriété suffisante pour améliorer la probabilité qu'un couple de messages forme une collision, il faut extraire la propriété permettant de caractériser ces couples, afin de construire efficacement le sous-espace de recherche. C'est à cette fin que les auteurs ont introduit la notion de bit neutre, dont voici la définition :

**Définition 5.10 (Bit neutre)**
*Soit $(M, M')$ un couple de messages se conformant à $\alpha_r$ pour un $r$ donné supérieur ou égal à 16, on dira que le bit $i$ ($i \in \{1, \ldots, 512\}$) est un bit neutre relativement au couple $(M, M')$ si le couple de messages obtenu en complémentant à 1 le bit $i$ de $M$ et de $M'$ se conforme aussi à $\alpha_r$.*

Cette notion peut être facilement étendue aux ensembles au moyen des définitions suivantes :

**Définition 5.11 (Paire neutre)**
*On dira que la paire de bit $(i, j)$ est une paire neutre relativement au couple $(M, M')$ si tous les couples de messages, obtenus en complémentant à 1 tout sous-ensemble de ces bits ($\{i\}$, $\{j\}$ et $\{i, j\}$) de $M$ et de $M'$, se conforment à $\alpha_r$.*

**Définition 5.12 (Ensemble neutre)**
*Un ensemble $E \subseteq \{1, \ldots, 512\}$, est neutre relativement au couple $(M, M')$ si pour tout sous-ensemble $F$ de $E$, tous les couples de messages obtenus en complémentant à 1 les bits correspondants à $F$ de $M$ et de $M'$ se conforment à $\alpha_r$.*

**Définition 5.13 (Ensemble 2-neutre)**
*Un ensemble $F \subseteq \{1, \ldots, 512\}$, est 2-neutre relativement au couple $(M, M')$ si chaque bit dans $F$ est neutre relativement au couple $(M, M')$ et chaque paire de bits dans $F$ est neutre relativement au couple $(M, M')$.*

**Construction du sous-espace de recherche de messages.** On construit le sous-espace de recherche à partir d'un couple de messages $(M, M')$ vérifiant la différentielle définie par le masque de perturbations $\mathcal{P}$ et les deux propriétés suivantes :

1. Le couple $(M, M')$ se conforme à $\alpha_r$.
2. Le couple $(M, M')$ possède un ensemble 2-neutre suffisamment grand. Les auteurs ont mesuré expérimentalement qu'environ 1/8 des sous-ensembles de bits de l'ensemble 2-neutre sont des ensembles neutres.

On note $k(r)$ le cardinal maximal de l'ensemble 2-neutre pour un couple $(M, M')$ et un $r$ donnés. Pour un couple de messages vérifiant ces deux propriétés, on peut construire un ensemble de $2^{k(r)}$ couples de messages en complémentant à 1 des sous-ensembles de bits de l'ensemble 2-neutre. Parmi ces couples, un nombre proche de $2^{k(r)-3}$ d'entre eux se conformeront à $\alpha_r$.

Soit $p_i$ la probabilité que le modèle de collisions locales se vérifie au pas $i$ (*i.e.* qu'il n'y ait pas de retenue lors de l'insertion d'une perturbation ou bien qu'une correction s'effectue convenablement), $p_i$ est égale à 1 s'il n'y a ni perturbation ni correction au pas $i$. La probabilité de vérification du modèle pour l'ensemble des 80 pas peut alors s'exprimer de la façon suivante :

$$p = \prod_{i=1}^{80} p_i.$$

Si les cinq dernières coordonnées du vecteur de perturbations sont nulles alors $p$ est la probabilité d'obtenir une collision, sinon c'est la probabilité d'obtenir une quasi-collision.

Pour un couple de message se conformant à $\alpha_r$ on note :

$$p(r) = \prod_{i=r+1}^{80} p_i.$$

Le nombre attendu de couple de messages se conformant à $\alpha_r$ à tester, pour en obtenir un qui vérifie le modèle pour les 80 pas, est $1/p(r)$. Pour construire le sous-espace de recherche, on choisira donc $r$ tel que $2^{k(r)}$ soit supérieur ou égal à $1/p(r)$. La valeur du paramètre $r$ choisie dans l'article de Biham et Chen est égale à 23.

**Construction d'un ensemble 2-neutre.** La construction d'un ensemble 2-neutre se déroule en deux étapes : on construit tout d'abord un ensemble de bits neutres, puis on identifie dans cet ensemble les paires neutres afin d'obtenir un sous-ensemble 2-neutre.

1. On choisit un couple de messages $(M, M')$ tel que $M' = M \oplus \mathcal{P}$ et $(M, M')$ se conforme à $\alpha_r$. Soit $e^i$, $i \in \{1, \ldots, 512\}$, le message dont le bit $i$ est égal à 1 et dont tous les autres bits sont égaux à 0. A partir du couple $(M, M')$, on construit 512 nouveaux couples de la forme :

   $$(M \oplus e^i, M' \oplus e^i) \quad \forall i \in \{1, \ldots, 512\}.$$

   On teste chaque couple, afin de déterminer s'il se conforme à $\alpha_r$. Si un couple se conforme à $\alpha_r$, alors le bit $i$ est un bit neutre. On obtient donc un ensemble de paires neutres relativement au couple $(M, M')$.

2. Pour construire un ensemble 2-neutre relativement au couple $(M, M')$, on définit un graphe dont les sommets correspondent aux bits neutres trouvés. On ajoute une arête pour chaque paire de bits neutres qui constitue une paire neutre. La clique maximale (le plus grand sous-ensemble de sommets tels que chacun des sommets de ce sous-ensemble est relié à tous les autres sommets du sous-ensemble) pour ce graphe constitue le plus grand ensemble 2-neutre

possible relativement au couple $(M, M')$. Biham et Chen précisent que si trouver la clique maximale d'un graphe est en général un problème NP-complet, dans le cadre de cette attaque, trouver une clique suffisamment grande (*i.e.* possédant au moins $k(r)$ sommets) n'est pas difficile car de nombreux sommets sont connectés à tous les autres.

**Construction d'un ensemble 2-neutre maximal.** Afin d'optimiser l'attaque, on souhaite trouver un moyen d'obtenir un couple de messages pour lequel l'ensemble 2-neutre relatif soit le plus grand possible. Pour ce faire, les auteurs décrivent des heuristiques tirées de résultats expérimentaux permettant de modifier le couple de départ $(M_0, M'_0)$ afin d'obtenir un nouveau couple susceptible d'être muni d'un ensemble 2-neutre plus grand. De plus, ils élargissent l'ensemble des paires neutres en considérant des paires, des triplés et des quadruplés de bits simultanément neutres, *i.e.* qui ne sont pas des bits neutres mais dont le couple de message obtenu en les complémentant à 1 simultanément se conforme à $\alpha_r$.

Une fois le nouveau couple $(M_1, M'_1)$ déterminé, on construit son ensemble 2-neutre. Si cet ensemble contient plus d'éléments que celui relatif au couple $(M_0, M'_0)$, on remplace $(M_0, M'_0)$ par $(M_1, M'_1)$, et on recommence jusqu'à obtenir un couple de messages muni d'un ensemble 2-neutre de taille suffisante.

**Conclusion.** La technique d'accélération de recherche de Biham et Chen a conduit à l'obtention des premières quasi-collisions pour la fonction de hachage SHA-0. Cependant bien qu'ayant prouvé son efficacité pour la cryptanalyse de SHA-0, la mise en oeuvre de cette technique pour la fonction SHA-1 est relativement problématique. En effet afin de se conformer à la caractéristique linéaire, les messages expansés correspondant au couple de messages candidats à la collision doivent vérifier un certain nombre de contraintes (évoquées section 5-3.4 de ce chapitre). Les couples de messages supplémentaires obtenus grâce à la technique des bits neutres doivent eux aussi vérifier ces contraintes sous peine de perdre complètement le fil de la caractéristique linéaire. Dans le cas de SHA-0, ces contraintes sont confinées aux positions de bit 1, 6 et 31 ; ces positions sont alors inéligibles en tant que bit neutre. Dans le cas de SHA-1, du fait de la permutation circulaire introduite dans la formule d'expansion, les positions de bit sujettes à des contraintes sont plus nombreuses ce qui réduit d'autant les choix de positions possibles pour les bits neutres. Finalement, cette technique d'accélération de recherche s'avère inefficace pour la fonction SHA-1.

### 5-6.2 Modifications de message

Les techniques de modifications de message furent initialement proposées par Wang *et al.* pour la cryptanalyse de la fonction MD5 [WY05]. Ces techniques furent ensuite adaptées et appliquées aux fonctions SHA-0 et SHA-1. Dans l'article [WYY05b], les auteurs font la distinction entre d'une part modifications de message simples et modifications de message avancées. Naito *et al.* [NSS06] ont proposé un nouveau type de modifications de message avancées baptisées modifications sous-marines (*Submarine Message Modifications*).

Le principe d'une modification de message consiste à modifier précisément certains bits ou combinaisons de bits des mots de message de telle sorte qu'une condition sur les bits des registres internes auparavant non vérifiée soit validée. Le processus

d'application des modifications de message se déroule pas à pas. Il en découle que les modifications de message utilisées à un certain pas ne doivent pas perturber les éventuelles modifications et conditions vérifiées par les états des registres précédents. On doit de plus vérifier que chaque modification de message s'accorde avec le système d'équations linéaires hérité de la construction et de l'instanciation de la caractéristique linéaire. La construction et la mise en oeuvre de ces modifications de message prend généralement place après qu'un chemin différentiel complet (vecteur de perturbations et caractéristique non-linéaire) ait été obtenu.

**Modification de message simple.** Les modifications de message simples interviennent seulement lors des 16 premiers pas de la fonction de mise à jour des registres, qui sont les pas pour lesquels l'attaquant possède un contrôle direct sur les mots de message. Ces modifications se résument en général à la modification d'un bit du mot de message correspondant à la même position de bit pour le registre. Pour modifier la valeur du bit $a_{i,j}$ (bit de position $j$ du registre $A_i$), on peut recalculer la valeur du mot de message $W_{i-1} = W_{i-1} + 2^j$.

On peut assez facilement s'assurer que l'ensemble des modifications de message simples permettent de corriger les conditions imposées aux états des registres des 16 premiers pas avec une probabilité de succès égale à 1. Une méthode triviale permettant de s'assurer de vérifier l'ensemble des conditions sur les registres $A_i$ pour $1 \leq i \leq 16$ consiste à utiliser la procédure suivante :

- Générer un ensemble d'états $A_i$ satisfaisant les conditions correspondant au chemin différentiel,
- Calculer $W_{i-1} = A_i - (A_{i-1} \lll 5) - IF(A_{i-2}, A_{i-3} \lll 30, A_{i-4} \lll 30) - (A_{i-5} \lll 30) - K_i$.

**Modification de message avancée.** Les modifications de message avancées visent à corriger des conditions non vérifiées par les bits des registres pour les pas supérieurs ou égaux à 16 où les mots de message utilisés ne sont plus sous le contrôle direct du cryptanalyste, mais issus du processus d'expansion. Ces modifications sont beaucoup plus complexes à construire. Leur utilisation repose le plus souvent sur la modification simultanée de plusieurs bits de message et s'appuient aussi sur la diffusion des modifications sur les états des registres.

Par exemple, afin de s'assurer de la condition $a_{21,4} = a_{20,4}$ que l'on peut trouver dans le chemin différentiel présenté dans [WYY05b], Naito *et al.* ont proposé la modification de message avancée qui consiste à ajouter les conditions supplémentaires suivantes :

$$\begin{aligned} a_{6,6} &= m_{5,6}, \\ m_{6,11} &\neq m_{5,6}, \\ m_{7,6} &= m_{5,6}, \\ a_{7,4} = 0, a_{8,4} &= 1, \\ m_{10,4} &\neq m_{5,6} \end{aligned}$$

puis à effectuer les modifications

$$M_5 = M_5 \oplus 2^5,$$
$$M_6 = M_6 \oplus 2^{10},$$
$$M_7 = M_7 \oplus 2^5,$$
$$M_{10} = M_{10} \oplus 2^3.$$

À la différence des modifications de message simples, des calculs de retenues interviennent dans les modifications de message avancées, il existe donc une probabilité que ces modifications échouent.

**Conclusion.** Les modifications de message avancées permettent de satisfaire des conditions sur les états des registres à des pas supérieurs au pas 16. Elles sont donc susceptibles d'induire une amélioration de la complexité de l'attaque, dès lors que leur probabilité d'échouer est petite. Des modifications de message ont été proposées pour corriger des conditions à hauteur du pas 24 pour SHA-0 [NSS06] et du pas 25 pour SHA-1 [WYY05c]. Cependant, ces modifications sont construites pour un chemin différentiel donné et dépendent fortement du vecteur de perturbations choisi. De plus, les modification publiées à ce jour ont été établies *à la main* ce qui rend fastidieuse leur utilisation en tant que technique d'accélération de recherche générique. Elles consomment aussi un grand nombre de degrés de liberté.

### 5-6.3 Boomerangs

L'attaque boomerang, introduite par Wagner [Wag99], est une attaque adaptative utilisant des textes clairs et chiffrés (*Adaptive chosen plaintext and ciphertext attack*). Elle fut développée ensuite par Kelsey *et al.* [KKS00] en attaque à clair choisi (*Choosen plaintext attack*) et rebaptisée attaque boomerang amplifiée. L'application des attaques boomerang amplifiées aux fonctions de hachage, et plus particulièrement à la fonction SHA-1, est due à Antoine Joux et Thomas Peyrin [JP07].

**Application aux schémas de chiffrement par bloc** L'attaque boomerang se fonde sur l'attaque différentielle. Son principe consiste à joindre deux caractéristiques différentielles courtes possédant une probabilité élevée en un quartet, afin d'obtenir un distingueur susceptible de couvrir plus de tours avec une meilleure probabilité. Afin de décrire l'attaque boomerang et l'attaque boomerang amplifiée, nous reprendrons ici les notations et les arguments de l'article de Kelsey *et al.* [KKS00]. Considérons un schéma de chiffrement par bloc $E$ décomposable en deux parties $e_0$ et $e_1$. Nous utiliserons les notations suivante pour décrire le chiffrement :

- $E(X_i) = e_1(e_0(X_i))$, avec
- $X_i$ : texte clair
- $Y_i = e_0(X_i)$,
- $Z_i = e_1(Y_i)$ : texte chiffré.

Supposons à présent que nous disposions de différentielles pour $e_0$, $e_0^{-1}$ et $e_1^{-1}$, et prenons de plus l'hypothèse que ces différentielles possèdent une probabilité égale à 1. Cette hypothèse ne sert qu'à simplifier au maximum la description de l'attaque

qui fonctionne pour des différentielles ayant des probabilités inférieures mais aussi pour des différentielles tronquées. Nous avons donc les différentielles suivantes :

$$\Delta_0 \longrightarrow \Delta_1 \text{ qui se propagent à travers } e_0 \text{ et } e_1.$$

Nous pouvons dès lors utiliser l'attaque boomerang de Wagner pour construire un distingueur pour le schéma de chiffrement $E$ de la façon suivante :

1. Faire la requête en chiffrement par $E$ de la paire de message $(X_0, X_1)$, vérifiant $X_0 \oplus X_1 = \Delta_0$.
2. Après $e_0$, la paire a été chiffrée en $(Y_0, Y_1)$ qui vérifie $Y_0 \oplus Y_1 = \Delta_1$. Après $e_1$, on obtient la paire $(Z_0, Z_1)$ pour laquelle il n'existe pas a priori de relation différentielle.
3. Nous construisons une nouvelle paire $(Y_2, Y_3)$ pour $e_1^{-1}$ à partir de la requête en déchiffrement par $E^{-1}$ de $Z_2 = Z_0 \oplus \Delta_1$ et $Z_3 = Z_1 \oplus \Delta_1$.
4. La paire $(Z_2, Z_3)$ est déchiffrée en $(Y_2, Y_3)$ avec les relations $Y_2 \oplus Y_0 = \Delta_0$ et $Y_3 \oplus Y_1 = \Delta_0$.
5. Cela détermine la relation différentielle entre $Y_2$ et $Y_3$ :

$$Y_0 \oplus Y_1 = \Delta_1, Y_2 \oplus Y_0 = \Delta_0, Y_3 \oplus Y_1 = \Delta_0 \text{ et donc } Y_2 \oplus Y_3 = \Delta_1.$$

6. Et comme $Y_2 \oplus Y_3 = \Delta_1$, nous avons une paire valide pour $e_0$ qui se propage pour donner $X_2 \oplus X_3 = \Delta_0$.

La probabilité qu'un tel évènement se produise pour une permutation aléatoire de taille $n$ bits est égale à $2^{-n}$. Il en résulte que cette procédure peut être utilisée pour distinguer un schéma de chiffrement par bloc $E$ d'une permutation aléatoire. L'attaque de Wagner nécessite à la fois des textes clairs choisis et des textes chiffrés choisis.

L'amélioration proposée par Kelsey *et al.* consiste à transformer cette attaque en attaque à clair choisi seulement. Supposons que le schéma de chiffrement par bloc $E$ traite des textes de taille $n$ bits. La requête de $2^{n/2}$ paires de textes clairs choisis $(X_{2i}, X_{2i+1})$ vérifiant $X_{2i} \oplus X_{2i+1} = \Delta_0$ nous fournit $2^{n/2}$ paires $(Y_{2i}, Y_{2i+1})$ vérifiant $Y_{2i} \oplus Y_{2i+1} = \Delta_1$, puisque nous avons émis l'hypothèse que les différentielles utilisées possèdent une probabilité égale à 1. De part le paradoxe des anniversaires, on peut s'attendre à l'existence dans cet ensemble d'une paire $(i, j)$ pour laquelle $Y_{2i} \oplus Y_{2j} = \Delta_0$. Dès que nous obtenons une telle paire, la propriété boomerang se vérifie :

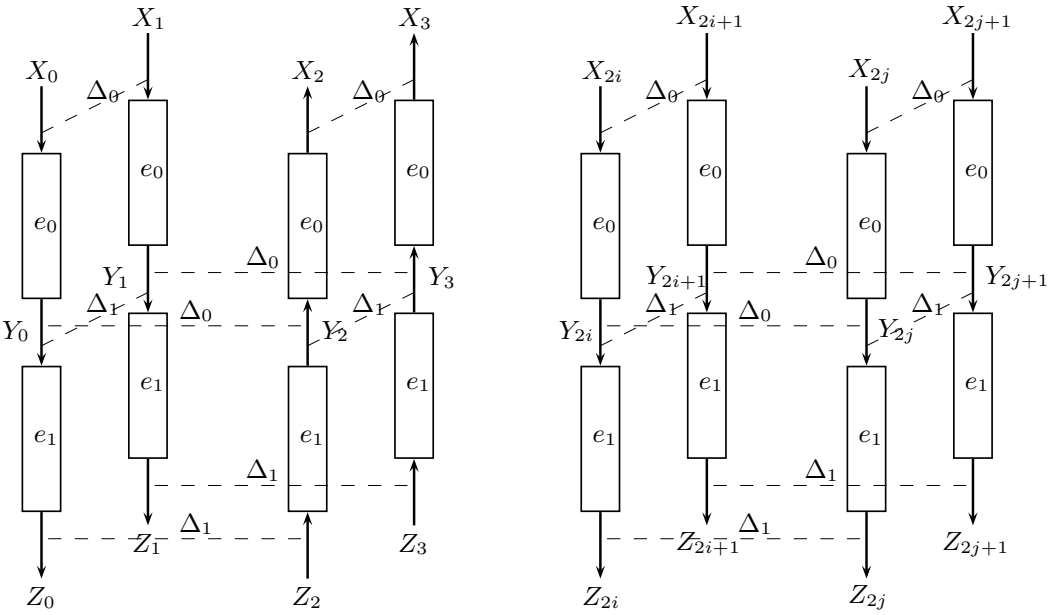
$$Y_{2i} \oplus Y_{2i+1} = \Delta_1, Y_{2j} \oplus Y_{2j+1} = \Delta_1, Y_{2i} \oplus Y_{2j} = \Delta_0 \text{ et donc } Y_{2i+1} \oplus Y_{2j+1} = \Delta_0.$$

De plus, nous avons aussi :

$$Z_{2i} \oplus Z_{2j} = \Delta_1 \text{ et } Z_{2i+1} \oplus Z_{2j+1} = \Delta_1.$$

Il existe environ $2^n$ paires $(i, j)$ possibles, et la probabilité pour qu'une paire donnée satisfasse les deux dernières équations est égale à $2^{-2n}$. Cette technique permet donc bien de distinguer le schéma de chiffrement $E$ d'une permutation aléatoire en utilisant seulement des textes clairs choisis.

L'avantage principal de ces attaques est qu'elles permettent de construire une attaque différentielle y compris pour des schémas de chiffrement pour lesquels il n'existe pas de bonne différentielle couvrant l'ensemble des rounds. Une illustration du principe de l'attaque boomerang et de l'attaque boomerang amplifiée pour les chiffrements par bloc se trouve figure 5-13.




Attaque boomerang. Attaque boomerang amplifiée.

FIG. 5-13 – Attaque boomerang et attaque boomerang amplifiée pour les schémas de chiffrement par bloc. Le sens des flèches indique le sens chiffrement/déchiffrement.

**Application aux fonctions de hachage** L'application de l'attaque boomerang amplifiée aux fonctions de hachage consiste à combiner à la différentielle principale plusieurs différentielles partielles, dites différentielles auxiliaires. Ces différentielles auxiliaires constituent des différentielles qui possèdent un bon comportement sur un nombre limité de pas de la fonction de mise à jour des registres.

Nous allons à présent décrire le principe d'application de l'attaque boomerang amplifiée pour les fonctions SHA-0 et SHA-1. Dans leur article, Joux et Peyrin [JP07] indiquent que cette technique constitue une généralisation des techniques de bits neutres et de modifications de message. Nous choisissons ici de nous placer dans le contexte d'une utilisation des boomerangs en tant que bits neutres, car c'est celle que nous avons utilisée avec succès lors de la cryptanalyse que nous avons menée contre la fonction SHA-0 [MP08].

Considérons une recherche de collision, les premières étapes de l'attaque nous ont permis de construire un chemin différentiel, que nous qualifierons de principal, et d'obtenir une différentielle $\nabla_P$, à partir de laquelle nous pouvons déduire la différence $\Delta_{Pw}$ appliquée en entrée au couple de messages (par simplicité, et sans perte de généralité, nous considérerons que toutes les différences utilisent le *ou exclusif* bit à bit). Dans les notations que nous emploierons, l'indice $w$ indique une différence concernant les mots de message, l'indice $a$ indique une différence concernant l'état des registres. Nous possédons aussi pour les états des registres, une différence en entrée $\Delta_{Ea}$ et une différence en sortie $\Delta_{Pa}$. Ces différences pourront être nulles dans le cas d'une recherche de collision sur un seul bloc, ou égale à des valeurs particulières dans le cas où la technique des blocs multiples est employée. Nous recherchons alors les paires de messages $(M, M')$ avec $M' = M \oplus \Delta_{Pw}$ pour lesquelles la différence de

sortie $\Delta_{Pa}$ est atteinte, ce qui se produit avec la probabilité $p_{\nabla P}$ pour chaque paire candidate.

Afin d'illustrer le principe de l'attaque boomerang, nous supposons que nous avons à notre disposition une différentielle auxiliaire $\nabla_A$ pour laquelle la différence appliquée aux messages vaut $\Delta_{Aw}$ et la différence observée sur l'état interne vaut $\Delta_{Aa}$. Une différentielle auxiliaire se caractérise de plus par sa "*portée*" $s$, c'est à dire le pas auquel la différence $\Delta_{Aa}$ est destinée à être obtenue. Pour une paire de message $(M, M')$ avec $M' = M \oplus \Delta_{Aw}$, la différence $\Delta_{Aa}$ est atteinte à la portée $s$ avec la probabilité $p_{\nabla A}$ pour chaque paire candidate. On peut remarquer que la différentielle principale $\nabla_P$ constitue une différentielle auxiliaire de portée 80.

Nous divisons l'ensemble des 80 pas de la fonction de mise à jour des registres en deux parties, la division étant positionnée sur le pas $s$ correspondant à la portée de la différentielle auxiliaire. Nous notons $\Delta_{Sa}$ la différence attendue sur l'état interne au pas $s$. La probabilité qu'une paire de messages $(M, M' = M \oplus \Delta_{Pw})$ conduise à un état interne vérifiant la différence $\Delta_{Sa}$ au pas $s$ est notée $p_S$. La probabilité qu'à partir d'une différence $\Delta_{Sa}$ sur l'état interne au pas $s$, une paire de message $(M, M' = M \oplus \Delta_{Pw})$ conduise à une différence $\Delta_{Pa}$ sur l'état interne au pas 80 est notée $p_F$. En supposant les étapes indépendantes, nous avons $p_{\nabla P} = p_S \times p_F$.

Une application de la technique des boomerangs amplifiées se déroule alors de la façon suivante :

1. On choisit une paire de messages $(M_1, M_1' = M_1 \oplus \Delta_{Pw})$ se conformant au chemin différentiel principal jusqu'au pas $s$, ceci est vrai avec la probabilité $p_S$.
2. On construit alors les paires $(M_1, M_2 = M_1 \oplus \Delta_{Aw})$ et $(M_1', M_2' = M_1' \oplus \Delta_{Aw})$, ces paires se conforment au chemin différentiel auxiliaire avec la probabilité $p_{\nabla A}$.

Si ces paires sont valides, la paire $(M_2, M_2')$ se conforme aussi au chemin différentiel principal jusqu'au pas $s$. La probabilité que cette paire se conforme à l'intégralité du chemin différentiel principal est égale à $p_{\nabla A}^2 \times p_F$. Cette technique permet donc, dès lors qu'une première paire de message $(M_1, M_1')$ se conformant au chemin différentiel principal jusqu'au pas $s$ est obtenue, de construire une seconde paire $(M_2, M_2')$ se conformant elle aussi jusqu'au pas $s$ avec une probabilité égale à $p_{\nabla A}^2$. En pratique, on peut construire relativement facilement, plusieurs chemins différentiels auxiliaires $\nabla A_i$ dont les probabilités $p_{\nabla A_i}$ sont très proches de 1. Il en résulte qu'à partir de $i$ chemins différentiels auxiliaires, il est possible d'amplifier une paire de messages conforme au pas $s$ en $2^i$ paires conformes jusqu'à ce même pas. L'utilisation de cette technique permet donc une accélération de la recherche d'une paire de messages valide. Une illustration du principe de l'attaque boomerang amplifiée se trouve figure 5-14.

Notre objectif ici étant de donner seulement une description concise de la technique des boomerangs amplifiée appliquée aux fonctions de hachage, nous invitons le lecteur intéressé par une description plus complète et détaillée à se référer à la thèse de Thomas Peyrin [Pey08].

**Conclusion** Les différentielles auxiliaires employées dans les attaques boomerangs constituent une généralisation des deux techniques d'accélération de recherche précédentes. Elles peuvent être utilisées de façon similaire aux bits neutres afin de multiplier les instances de paires de message conformes, ou bien de façon similaire au

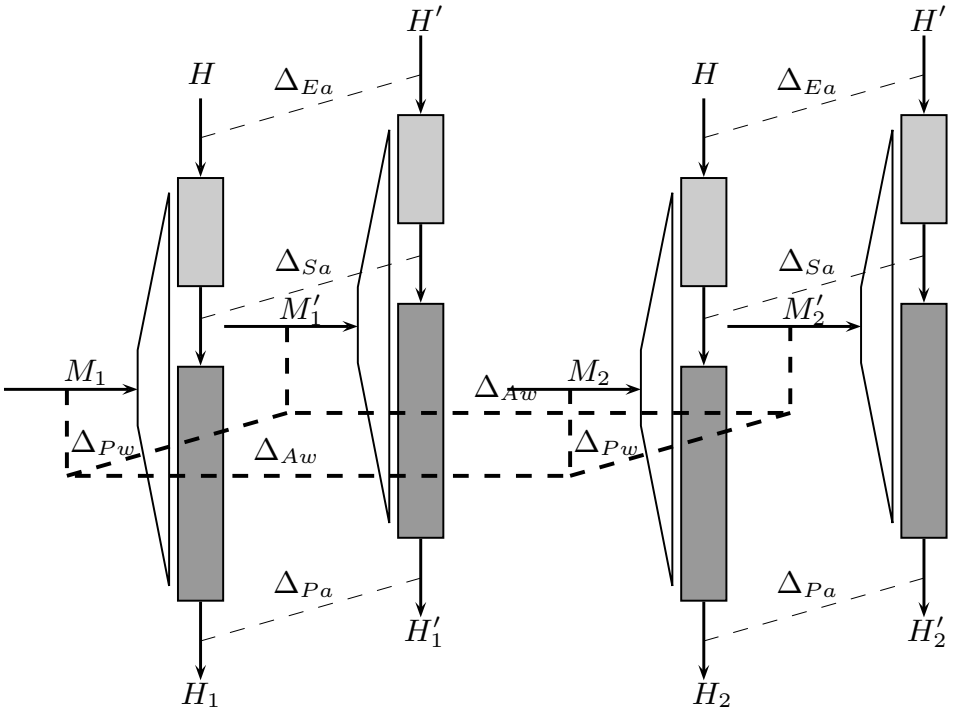



FIG. 5-14 – Attaque boomerang amplifiée appliquée aux fonctions SHA-0 et SHA-1.

modifications de message afin de modifier des paires de message pour amener des conditions portant sur des bits de registres à être vérifiées. Elles constituent à ce jour, la technique d'accélération de recherche la plus efficace pour la cryptanalyse des fonctions SHA-0 et SHA-1.

# 6

# Accélération de la cryptanalyse de SHA-0

**Sommaire**



## 6-1 Introduction

La première attaque par collision contre la fonction SHA-0 fut proposée par Chabaud et Joux en 1998 [CJ98]. Cet article se concentre essentiellement sur la caractéristique linéaire, et 3 contraintes sur le vecteur de perturbations sont imposées afin d'obtenir une collision sur un seul bloc. Le vecteur proposé conduisait à une probabilité de succès de l'ordre de $2^{-68}$. Cependant, un implémentation astucieuse de l'attaque permettait de réduire la complexité de l'attaque à $2^{61}$ évaluations de la fonction de compression. La contribution de l'article de Biham et Chen de 2004 [BC04] apporta la première technique d'accélération de recherche sous la forme des bits neutres. Puis l'ajout de l'application de la technique des blocs multiples dans [BCJ05] permit d'obtenir la première collision pour la fonction SHA-0 avec une complexité de $2^{51}$ appels à la fonction de compression. L'attaque de 2005 de Wang *et al.* [WYY05d], se fondit sur une nouvelle façon d'appréhender la différentielle et d'envisager sa façon de se propager au sein des registres. Forts de cette approche, les auteurs relâchèrent 2 des trois contraintes imposées au vecteur de perturbations et mirent à jour un nouveau vecteur plus efficace. Cependant, c'est l'introduction d'un nouvel outil dénommé caractéristique non-linéaire qui autorisa la mise en oeuvre de ce vecteur et permit d'abaisser la complexité de l'attaque à $2^{39}$ appels de la fonction de compression. Naito *et al.* [NSS06] apportèrent leur contribution à l'édifice en proposant de nouvelles pistes pour la phase d'accélération de recherche, conduisant à une attaque théorique munie d'une complexité de l'ordre de $2^{36}$ appels.

Finalement, la meilleure attaque publiée à ce jour est issue d'un travail commun avec Thomas Peyrin [MP08], et possède une complexité de l'ordre de $2^{33}$ évalua-

tions de la fonction de compression. C'est cette attaque que nous présentons dans ce chapitre.

## 6-2 Nouveau vecteur de perturbations

La caractéristique linéaire utilisée dans l'attaque de Wang *et al.* a pour objectif l'obtention de collisions sur un seul bloc : d'où la contrainte sur les cinq dernières coordonnées de leur vecteur de perturbations. On peut d'ailleurs remarquer que même si la collision exhibée dans l'article [WYY05d] utilise deux blocs de message, elle ne constitue pas au sens strict une utilisation de la technique des blocs multiples. En effet, les variables de chaînage issues de la première itération de la fonction de compression sont identiques et obtenues avec un même bloc de message. L'utilisation de ce premier bloc ne s'avère nécessaire que pour obtenir une variable de chaînage dont les bits vérifient un certain nombre [2] d'équations héritées de la caractéristique non-linéaire.

Nous avons choisi de construire notre attaque en tirant pleinement partie de la technique des blocs multiples, ce qui permet de relâcher la dernière contrainte pesant sur le vecteur de perturbations. Une fois toutes les contraintes relâchées, nous avons défini comme critère pour évaluer les vecteurs candidats de minimiser le nombre de conditions à satisfaire entre les étapes 17 et 80. Nous avons donc énuméré exhaustivement l'ensemble des vecteurs de perturbations selon ce critère de sélection. Nous avons trouvé de nombreux vecteurs présentant un nombre de conditions aux alentours de 40. Cependant, nous avons remarqué que ces vecteurs présentaient un comportement inégal relativement à l'outil de génération de caractéristique linéaire et à la technique d'accélération de recherche fondée sur les boomerangs. En conséquence, nous avons construit la caractéristique non-linéaire pour chacun des vecteurs les plus prometteurs pour finalement retenir le vecteur présenté dans le tableau 6-1.

Le nombre total de conditions entre les pas 17 et 80 pour ce vecteur s'élève à 42. Cependant, on peut facilement s'affranchir des conditions présentes au pas 17 de la même façon que dans [CJ98]. De plus en pratique, lors de notre recherche effective de collision, une "mauvaise" paire de message est détectée très rapidement (après environ 20 pas en moyenne). Il en résulte que la complexité théorique, en nombre d'évaluation de la fonction de compression de SHA-0 et avant utilisation d'une technique d'accélération de recherche, est égale à $2^{40}/2^2 = 2^{38}$, ce qui a été confirmé par l'implémentation pratique ($2^{37,9}$ et $2^{37,8}$ évaluations de SHA-0 en moyenne pour le premier et le second bloc respectivement).

## 6-3 Utilisation des boomerangs

Nous avons appliqué à la fonction SHA-0 la technique d'accélération de recherche des boomerangs pour les fonctions de hachage introduite dans [JP07]. Pour notre attaque, nous avons utilisé deux types différents de différentielles auxiliaires mises en oeuvre dans le contexte des bits neutres.

Nous avons défini la portée d'une différentielle auxiliaire comme étant le dernier pas où les différences incontrôlées issues de cette différentielle n'interfèrent pas avec

[2] L'article de Wang *et al.* fait mention de 14 conditions cependant le nombre exact de conditions est 16 [NSS06, Man06].

| Pas 1 to 20 | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| vecteur | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| ♯ conditions | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 2 | 0 | 2 | 1 |

| Pas 21 to 40 | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| vecteur | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| ♯ conditions | 1 | 0 | 2 | 0 | 2 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |

| Pas 41 to 60 | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| vecteur | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| ♯ conditions | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 2 | 1 | 2 | 2 |

| Pas 61 to 80 | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| vecteur | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| ♯ conditions | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |

TAB. 6-1 – Nouveau vecteur de perturbations pour SHA-0, avec le nombre de conditions pour chaque pas (les conditions des 16 premiers ne sont pas prises en compte).

le chemin différentiel principal. Les deux différentielles auxiliaires que nous avons utilisées sont notées $AP_1$ et $AP_2$ :

- la différentielle auxiliaire $AP_1$ présente peu de contraintes et possède une portée faible,
- la seconde différentielle $AP_2$ nécessite de plus nombreuses contraintes, mais possède une portée supérieure.

Un compromis a été fait entre ces deux types de différentielles afin d'obtenir le meilleur ensemble possible de chemins auxiliaires sans toutefois trop contraindre le chemin différentiel principal, ce qui pourrait avoir comme effet de faire échouer le processus de génération de la caractéristique non-linéaire.

**Collisions locales non-linéaires.** La différentielle associée à une collision locale se présente sous la forme suivante :

$$\Delta =< 2^j, 2^{j+5}, 2^j, 2^{j+30}, 2^{j+30}, 2^{j+30} > .$$

La perturbation insérée à l'étape 1 sur une position de bit $j$ est successivement corrigée dans les étapes suivantes par l'injection de perturbations sur les positions de bit $j+5$ à l'étape 2, $j$ à l'étape 3, $j+30$ aux étapes 4, 5 et 6. Les étapes 3, 4 et 5 correspondent aux cas où la perturbation initiale se propage à travers la fonction de ronde. Cette séquence de corrections est construite pour conduire à l'obtention d'une collision sur 6 pas dans le modèle linéaire. Il s'agit de la séquence présentant les meilleures probabilités de succès pour l'ensemble des 80 pas de la fonction de mise à jour des registres, dont la moitié utilise la fonction $XOR$. Cependant, si nous choisissons de nous restreindre au cas de la fonction $IF$, il existe d'autres séquences de corrections susceptibles de mener à l'obtention d'une collision sur 6 pas. En effet, la fonction $IF$ possède la propriété de pouvoir absorber les différences avec une probabilité 1/2. Nous pouvons donc choisir, lors des 3 étapes où la perturbation initiale est susceptible de se propager à travers la fonction de ronde, d'introduire ou

| pas | $\nabla_{A_i}$ | $\nabla_{W_i}$ |
|---|---|---|
| | ... | ... |
| $i-1$ | ----------b--- | ---------- |
| $i$ | ----------b--- | ----------a- |
| $i+1$ | ----------a- | ----------$\overline{\text{a}}$------ |
| $i+2$ | 0---------- | ---------- |
| $i+3$ | 1---------- | ---------- |
| $i+4$ | ---------- | ---------- |
| $i+5$ | ---------- | $\overline{\text{a}}$---------- |
| $i+6$ | ---------- | ---------- |
| | ... | ... |

FIG. 6-1 – Caractéristique correspondant à une collision locale non-linéaire initiée au pas $i$ et de différentielle $\Delta =< 2^1, 2^6, 0, 0, 0, 2^{31} >$. La colonne de droite représente les différents états des mots de message expansés. La colonne de gauche représente les différents états du registre $A$ et précise les contraintes nécessaires pour que la collision locale non-linéaire soit vérifiée. Un "-" indique qu'il n'y a pas de contrainte, une lettre indique une valeur binaire et une lettre surlignée son complément.

non une correction. Il en découle que par exemple la différentielle

$$\Delta =< 2^j, 2^{j+5}, 0, 0, 0, 2^{j+30} >$$

peut être utilisée pour obtenir une collision sur 6 pas lorsque la fonction de ronde est la fonction $IF$. Pour la position de bit $j = 1$, cette différentielle possède d'ailleurs une meilleure probabilité de succès ($2^{-4}$) que la différentielle classique ($2^{-5}$). Nous associons le nom de collisions locales non-linéaires à ce type de différentielle. Nous pouvons choisir d'effectuer ou de ne pas effectuer de correction pour les étapes 4, 5 et 6 ; nous pouvons donc construire $2^3 = 8$ collisions locales non-linéaires différentes. Ces collisions non-linéaire ont pour vocation d'intervenir lors des premiers pas de la fonction de mise à jour des registres là ou l'attaquant possède un contrôle direct sur les mots de message et donc sur les registres. Il est donc possible de contrôler précisément les conditions suffisantes pour qu'une collision locale non-linéaire se vérifie avec une probabilité égale à 1. La figure 6-1 donne un exemple de représentation d'une collision locale non-linéaire avec les conditions suffisantes associées.

**Construction des différentielles auxiliaire $AP_1$ et $AP_2$.** Le principe d'une différentielle auxiliaire consiste à construire, au moyen de collisions locales non-linéaires, un chemin différentiel partiel conduisant à une collision à un pas déterminé. Étant donné que pour cette attaque nous avons choisi de nous placer dans le contexte d'une utilisation des différentielles auxiliaires en tant que bits neutres, nous souhaitons que ces différentielles possèdent la plus grande portée possible. Cependant ce chemin doit obéir à un certain nombre de contraintes :

- Afin de bénéficier du processus de compensation hérité de l'expansion de message de SHA-0, nous avons choisi de positionner les collisions locales non-linéaires d'une même différentielle auxiliaire sur la même position de bit.
- Pour une collision locale non-linéaire débutant au pas $i$, des conditions doivent être vérifiées (et donc imposées) sur des bits du registre $A_{i+2}$. Nous n'autorisons

donc l'apparition de collision locale non-linéaire que sur les mots $W_0, \ldots, W_{14}$, car au delà il n'est théoriquement plus possible de s'assurer d'une probabilité égale à 1 pour la différentielle auxiliaire.

Finalement, le nombre de candidat possible est égal à $8^{15} = 2^{45}$. En pratique, nous imposons une condition supplémentaire sur le nombre de collisions locales non-linéaire que nous avons limité à 5 pour notre recherche. En effet si ce nombre est trop élevé, trop de degrés de liberté sont dépensés. Pour chaque candidat, on applique la formule de récurrence de l'expansion de message et on mesure la portée correspondant à ce candidat en identifiant le pas où la première différence incontrôlée apparaît. Parmi les candidats obtenus, nous avons retenus deux différentielles.

La différentielle $AP_1$ utilise une unique collision locale non-linéaire de la forme $\Delta =< 2^j, 2^{j+5}, 0, 0, 0, 2^{j+30} >$. Cette collision est initiée au pas 7, et la première différence incontrôlée apparaît au pas 20. La figure 6-2 donne les 32 premières étapes correspondantes à l'expansion de message associée à la différentielle auxiliaire $AP_1$. Les conditions devant être vérifiées afin d'obtenir une probabilité de succès égale à 1 pour la différentielle $AP_1$ sont données dans la figure 6-3.

La différentielle $AP_2$ utilise un enchevêtrement de 3 collisions locales non-linéaires initiées au pas 1 (forme $\Delta =< 2^j, 2^{j+5}, 0, 2^{j+30}, 2^{j+30}, 2^{j+30} >$), ainsi qu'aux pas 3 et 11 (forme $\Delta =< 2^j, 2^{j+5}, 0, 0, 2^{j+30}, 2^{j+30} >$). Cette configuration permet de faire apparaître la première différence incontrôlée au pas 25. La figure 6-4 donne les 32 premières étapes correspondantes à l'expansion de message associée à la différentielle auxiliaire $AP_2$. Les conditions devant être vérifiées afin d'obtenir une probabilité de succès égale à 1 pour la différentielle $AP_2$ sont données dans la figure 6-5.

Conformément à la remarque présente dans l'article de Joux et Peyrin, nous notons qu'une différence auxiliaire possédant une portée $s$, constitue un bit neutre valide pour le pas $s + 3$ avec une très forte probabilité (la différence incontrôlée doit se propager avant d'interférer avec le chemin différentiel principal). Il résulte de cette remarque que les différentielles $AP_1$ et $AP_2$ peuvent être utilisées en tant que bits neutres pour les pas 23 et 28 respectivement.

**Placement des différentielles auxiliaires.** Une fois les différentielles auxiliaires établies, nous disposons de plusieurs stratégies pour leur placement. La première de ces stratégies consiste à utiliser les différentielles auxiliaires en suivant la méthode de Biham et Chen pour les bits neutres. Dès qu'une paire de message se conforme à la différentielle principale jusqu'à la portée correspondant à une différentielle auxiliaire, on tente de multiplier les instances en essayant de placer cette différentielle auxiliaire. Cependant, cette stratégie est peu productive car elle nécessite de trouver des positions de bit pour lesquelles les conditions de la différentielle auxiliaire sont vérifiées. Une stratégie plus intéressante consiste à lier le placement des différentielles auxiliaires à la génération de caractéristique non-linéaire. Deux choix sont alors possibles, selon que l'on place les différentielles auxiliaires avant ou après la génération de la caractéristique non-linéaire. Si on place les différentielles auxiliaires après l'établissement de cette caractéristique, la probabilité de trouver des positions satisfaisantes est meilleure qu'avec la méthode de Biham et Chen. Mais le nombre de degrés de liberté pour le placement est limité par le nombre de bits non-contraints lors de la génération de la caractéristique non-linéaire. Le choix qui laisse le plus grand nombre de degrés de libertés, et qui est donc susceptible de conduire à la meilleure accélé-

| | $W_0$ à $W_{15}$ | $W_{16}$ à $W_{31}$ |
|---|---|---|
| Perturbations sur $W^j$ | 0000001000000000 | 0000101101100111 |
| Corrections sur $W^{j+5}$ | 0000000100000000 | 0000010110110011 |
| Corrections sur $W^{j+30}$ | 0000000000010000 | 000$\underline{1}$001000000010 |

FIG. 6-2 – Premières étapes de l'expansion de message associée à la différentielle auxiliaire $AP_1$. La collision locale non-linéaire est initiée au pas 7 ($W_6^j$). La première différence incontrôlée apparaît pas 20 ($W_{19}^{j+30}$).

| $i$ | $A_i$ | $W_i$ |
|---|---|---|
| 00 : | | ---------- |
| 01 : | ---------- | ---------- |
| 02 : | ---------- | ---------- |
| 03 : | ---------- | ---------- |
| 04 : | ---------- | ---------- |
| 05 : | ----------b---- | ---------- |
| 06 : | ----------b---- | ----------a-- |
| 07 : | ----------a-- | ----------$\overline{\text{a}}$------- |
| 08 : | ----------0 | ---------- |
| 09 : | ----------1 | ---------- |
| 10 : | ---------- | ---------- |
| 11 : | ---------- | ----------$\overline{\text{a}}$ |
| 12 : | ---------- | ---------- |
| 13 : | ---------- | ---------- |
| 14 : | ---------- | ---------- |
| 15 : | ---------- | ---------- |

FIG. 6-3 – Caractéristique associée à la différentielle auxiliaire $AP_1$ positionnée sur la position de bit $j = 1$. Cette différentielle possède une seule collision locale non-linéaire de la forme $\Delta =< 2^j, 2^{j+5}, 0, 0, 0, 2^{j+30} >$ initiée au pas 7. La colonne de droite représente les différents états des mots de message expansés. La colonne de gauche représente les différents états du registre $A$ et précise les contraintes nécessaires pour que la collision locale non-linéaire soit vérifiée. Un " - " indique qu'il n'y a pas de contrainte, une lettre indique une valeur binaire et une lettre surlignée son complément.

| | $W_0$ à $W_{15}$ | $W_{16}$ à $W_{31}$ |
|---|---|---|
| Perturbations sur $W^j$ | 1010000001000000 | 00000000$\underline{1}$0110110 |
| Corrections sur $W^{j+5}$ | 0101000000100000 | 0000000001011011 |
| Corrections sur $W^{j+30}$ | 0001111100000011 | 0000000000001110 |

FIG. 6-4 – Premières étapes de l'expansion de message associée à la différentielle auxiliaire $AP_2$. Les collisions locales non-linéaires sont initiées aux pas 1, 3 et 11 ($W_0^j$, $W_2^j$, $W_{10}^j$). La première différence incontrôlée apparaît pas 25 ($W_{24}^j$).

| $i$ | $A_i$ | $W_i$ |
|---|---|---|
| -1 : | ----------d---- | |
| 00 : | ----------d---- | ----------a-- |
| 01 : | ----------e-a-- | ----------$\overline{a}$------- |
| 02 : | ----------e---1 | ----------b-- |
| 03 : | ----------b-0 | ----------$\overline{b}$------$\overline{a}$ |
| 04 : | ----------0 | ----------$\overline{a}$ |
| 05 : | ----------0 | ----------$\overline{a}$ |
| 06 : | ---------- | ----------$\overline{b}$ |
| 07 : | ---------- | ----------$\overline{b}$ |
| 08 : | ---------- | ---------- |
| 09 : | ----------f---- | ---------- |
| 10 : | ----------f---- | ----------c-- |
| 11 : | ----------c-- | ----------$\overline{c}$------- |
| 12 : | ----------0 | ---------- |
| 13 : | ----------0 | ---------- |
| 14 : | ---------- | ----------$\overline{c}$ |
| 15 : | ---------- | ----------$\overline{c}$ |

FIG. 6-5 – Caractéristique associée à la différentielle auxiliaire $AP_2$ positionnée sur la position de bit $j = 1$. Cette différentielle possède une collision locale non-linéaire de la forme $\Delta =< 2^j, 2^{j+5}, 0, 2^{j+30}, 2^{j+30}, 2^{j+30} >$ initiée au pas 1, et deux collisions locales non-linéaires de la forme $\Delta =< 2^j, 2^{j+5}, 0, 0, 2^{j+30}, 2^{j+30} >$ initiées aux pas 3 et 11. La colonne de droite représente les différents états des mots de message expansés. La colonne de gauche représente les différents états du registre $A$ et précise les contraintes nécessaires pour que la collision locale non-linéaire soit vérifiée. Un "-" indique qu'il n'y a pas de contrainte, une lettre indique une valeur binaire et une lettre surlignée son complément.

| | Premier bloc | Second bloc |
|---|---|---|
| Différentielle | Positions de bit | Positions de bit |
| $AP_1$ | $j = \{9, 11\}$ | $j = \{9, 11\}$ |
| $AP_2$ | $j = \{10, 14, 19, 22, 27\}$ | $j = \{17, 22, 30\}$ |

TAB. 6-2 – Type des différentielles auxiliaires utilisées lors de la cryptanalyse et leurs positions respectives.

ration de recherche, consiste à placer les différentielles auxiliaires avant de tenter de générer une caractéristique non-linéaire. Cependant, la perte de degrés de libertés induite par le placement des différentielles auxiliaires peut rendre l'établissement de la caractéristique non-linéaire difficile voire même impossible.

C'est cette dernière stratégie que nous avons mise en oeuvre pour notre attaque. Le tableau 6-2 récapitule pour chaque bloc le nombre de différentielles auxiliaires utilisées, leur type et leur position de bit. Le nombre supérieur de différentielles $AP_2$ pour le premier bloc tient à la valeur de la variable de chaînage initiale définie pour la fonction SHA-0 qui est fortement structurée. Il s'avère que cette structure profite au cryptanalyste car elle permet d'introduire plus de chemins différentiels auxiliaires.

**Utilisation des différentielles auxiliaires.** L'utilisation des différentielles auxiliaires intervient lors de la phase de recherche effective d'une collision. Dès qu'une paire de messages se conforme au pas 23 (portée de la différentielle auxiliaire $AP_1$) on applique la différentielle $AP_1$ pour construire une seconde paire de messages. Cette nouvelle paire se conforme au pas 23 avec une probabilité très proche de 1. Le nombre de nouvelles instances obtenues est égale au nombre de différentielles $AP_1$ placées. On vérifie la conformité de la paire initiale et des nouvelles paires au pas 28 (portée de la différentielle auxiliaire $AP_2$). Pour toutes les instances conforme au pas 28 on déclenche la différentielle $AP_2$ pour multiplier les instances. On vérifie enfin pour toutes les instances restantes si l'une d'entre elle conduit à une collision.

Théoriquement, avec $k$ chemins différentiels auxiliaires, on peut s'attendre à une amélioration de la recherche de collisions par un facteur $2^k$. Bien que ce facteur d'amélioration soit fortement corrélé à la portée et au nombre de différentielles auxiliaires introduites. Nous avons utilisé 7 différentielles auxiliaires pour le premier bloc et 5 différentielles auxiliaires pour le second bloc. Le gain sur la recherche de collisions devrait donc être respectivement $2^7$ pour le premier bloc et $2^5$ pour le second bloc ; soit des complexités théoriques de $2^{39-7} = 2^{32}$ et $2^{38-5} = 2^{33}$. Les résultats obtenus en pratique sont proches de la théorie, avec une complexité mesurée égale à $2^{32,2}$ et $2^{33}$ évaluations de la fonction de compression, respectivement pour le premier et le second bloc.

## 6-4 Caractéristique non-linéaire

Les caractéristiques non-linéaires utilisées pour les deux blocs de la cryptanalyse ont été obtenues à l'aide d'un générateur automatique fondé sur les travaux de De

Cannière et Rechberger [DCR06]. Cependant, nous avons imposé comme contrainte supplémentaire au générateur de prendre en compte les différentielles auxiliaires nécessaires à l'accélération de l'attaque. En pratique pour une variable de chaînage aléatoire, il est relativement simple de trouver une caractéristique non-linéaire capable d'intégrer 5 différentielles auxiliaires, avec au moins 3 différentielles de type $AP_2$. Les tableaux 6-3 et 6-4 détaillent les chemins différentiels complets pour le premier bloc, les tableaux 6-5 et 6-6 pour le second bloc.

## 6-5 Conclusion

L'application de la technique des blocs multiples ainsi que l'utilisation des boomerangs nous ont permis de trouver une collision pour la fonction SHA-0 avec une complexité mesurée égale à $2^{33.6}$ évaluations de la fonction de compression. Cette cryptanalyse constitue, à ce jour, la meilleure attaque par collision publiée contre la fonction SHA-0. La figure 6-7 donne un exemple de collision obtenue lors de l'implémentation de notre attaque.

| $i$ | $\nabla_{A_i}$ | $\nabla_{W_i}$ |
|---|---|---|
| -4 : | 00001111010010111000011111000011 | |
| -3 : | 01000000110010010101000111011000 | |
| -2 : | 01100010111010110111001111111010 | |
| -1 : | 11101111110011011010101110001001 | |
| 00 : | 01100111010001010010001100000001 | 01001110010010110000001010n0010u1 |
| 01 : | 11101101111111111100111011n1111u0 | 01000000110110110101000001n000000 |
| 02 : | 01100111011111111101n10101101010 | 11110110000111110100111011n011011 |
| 03 : | 001101010010100n00001100n0uuuuu0 | 00111000101010010111001000u000101 |
| 04 : | 111100000nu0000010101100011100000 | u11001000100000000101000000u0101n1 |
| 05 : | 00111n0000001001100010000u0n1u1 | n00101001100100001010100101000000 |
| 06 : | 10110101110110110000101u100u1001 | n01000101111010000111100u000100 |
| 07 : | 100unnnnnnnnn0100nu0100101u11001 | 00010010111101000101011001011010 |
| 08 : | 1000011100001n000n100u0n010nn001 | 01011010011100010001100001n0001u1 |
| 09 : | 0010000000000010un00nu1u1un01100 | n10100010100011111001011000u1110n0 |
| 10 : | 11100110100101000nu01u10un00n100 | 01111010011000100100011100011000 |
| 11 : | 01111000111000110 1nuu10101000101 | 01110001001101110101100 11u1110u0 |
| 12 : | 01001101011010000010u0000n110000 | 10110111110101-----1-----u000001 |
| 13 : | 010110011100000----010-0-01001u0 | 101001010----------------n0000u1 |
| 14 : | 10111100--------------1--110u011 | 01101-0----0--1----0---0-1-011u0 |
| 15 : | 10100------------------0-1-u0100 | n0101-0----0--1----0---0-1-11000 |
| 16 : | --01----------------------n0011 | 010001110----------------00101n0 |
| 17 : | ----------------------------1n- | n1000-0----1--1----1---0-u-10011 |
| 18 : | 1---------------------------0-- | 01000-0----1--1----0---0-0-011u0 |
| 19 : | ------------------------------- | n00110100----------------0001011 |
| 20 : | ------------------------------- | n0110-0----1-------0-----0-000u1 |
| 21 : | ----------------------------n- | u1100-1------------------u-10111 |
| 22 : | ------------------------------- | 00001-1------------------0-00110 |
| 23 : | ----------------------------n- | n1011-1----0-------0-----u-11001 |
| 24 : | ------------------------------- | u0000-0------------------1-11100 |
| 25 : | ----------------------------n- | 01101-1------------------u-10111 |
| 26 : | ------------------------------- | u1010-1----0-------1-----0-011u0 |
| 27 : | ------------------------------- | 01001-1------------------0-01110 |
| 28 : | ------------------------------- | u0000-0------------------1-11011 |
| 29 : | ------------------------------- | u0111-0------------------0-00010 |
| 30 : | ------------------------------- | 01101-1------------------1-10010 |
| 31 : | ------------------------------- | 10110-1------------------0-01001 |
| 32 : | ------------------------------- | 00111-1------------------1-00100 |
| 33 : | ------------------------------- | 01011-1------------------1-11101 |
| 34 : | ------------------------------- | 00010-0------------------0-010u0 |
| 35 : | ----------------------------u- | 10001-0------------------n-10110 |
| 36 : | ------------------------------- | 11100-0------------------0-000u1 |
| 37 : | ------------------------------- | n0010-0------------------0-001u0 |
| 38 : | ----------------------------u- | n1101-0------------------n-11110 |
| 39 : | ------------------------------- | n1100-1------------------0-001n0 |
| 40 : | ------------------------------- | n1111-0------------------0-10000 |
| | ... | ... |

TAB. 6-3 – Chemin différentiel complet pour le premier bloc, pas 1 à 40.

| $i$ | $\nabla_{A_i}$ | $\nabla_{W_i}$ |
|---|---|---|
| | ... | ... |
| 41 : | ---------- | n0010-1----------0-11010 |
| 42 : | ---------- | n0100-0----------1-110u1 |
| 43 : | ----------u- | 00000-1----------n-01010 |
| 44 : | ---------- | 00011-0----------0-100n0 |
| 45 : | ---------- | n0111-1----------1-10110 |
| 46 : | ---------- | n0111-1----------0-00010 |
| 47 : | ---------- | u0010-1----------1-00000 |
| 48 : | ---------- | 01101-0----------0-010n0 |
| 49 : | ----------n- | 11111-1----------u-10011 |
| 50 : | ---------- | 01000-1----------0-100u0 |
| 51 : | ---------- | u1110-1----------0-10010 |
| 52 : | ---------- | n1101-1----------1-11110 |
| 53 : | ---------- | n0001-1----------1-001u0 |
| 54 : | ----------u- | 11011-0----------n-11110 |
| 55 : | ---------- | 10001-0----------0-000n0 |
| 56 : | ---------- | n0111-1----------0-001n1 |
| 57 : | ----------n- | n0110-1----------u-11101 |
| 58 : | ---------- | u1110-1----------1-11001 |
| 59 : | ----------n- | u1110-0----------u-010u1 |
| 60 : | ----------u- | n1111-1----------n-100n1 |
| 61 : | ---------- | 01010-0----------0-010n1 |
| 62 : | ---------- | 01111-1----------1-11111 |
| 63 : | ---------- | 10011-1----------0-00010 |
| 64 : | ---------- | n1000-0----------0-10110 |
| 65 : | ---------- | 01000-0----------1-00011 |
| 66 : | ---------- | 01000-0----------0-101u1 |
| 67 : | ----------u- | 01001-0----------n-01001 |
| 68 : | ---------- | 10001-0----------0-100u0 |
| 69 : | ---------- | u0010-1----------1-11000 |
| 70 : | ---------- | u1010-0----------1-011n1 |
| 71 : | ----------n- | u0101-0----------u-01101 |
| 72 : | ---------- | 00011-1----------0-100u0 |
| 73 : | ---------- | n1010-1----------0-11000 |
| 74 : | ---------- | n1100-0----------0-10010 |
| 75 : | ---------- | u1110-1----------1-110n1 |
| 76 : | ----------n- | 11011-1----------u-00100 |
| 77 : | ---------- | 00111-0----------1-000n1 |
| 78 : | ---------- | n0011-0----------1-11101 |
| 79 : | ---------- | u0101-0----------1-01000 |
| 80 : | ---------- | |

TAB. 6-4 – Chemin différentiel complet pour le premier bloc, pas 41 à 80.

| $i$ | $\nabla_{A_i}$ | $\nabla_{W_i}$ |
|---|---|---|
| -4 : | 11101101000010101000111010101 0u1 | |
| -3 : | 01000110011100010110101100101000 | |
| -2 : | 10010000011011111010001110100111 | |
| -1 : | 11100110001011011100010100001001 | |
| 00 : | 01011110101010111100001100111101 | 10011010011101101100111 10u1100n0 |
| 01 : | u0111011010011100010111unn1010n1 | 00000100101110011001010 10u111101 |
| 02 : | 11010011001110011011u000110u0111 | 11101110001001000010001 00n111101 |
| 03 : | 11100111100001 0u000unnnnnn000100 | 10011010010001100110011 01n110101 |
| 04 : | u0100101unn01010000u100011110110 | u0111000010000000000010 101u1101u0 |
| 05 : | n000un001000011u00100000000nn0n0 | u0111000011000000000011 0000010010 |
| 06 : | nnn0010001011110011100n1nu1u011u | u0001011011111101000010 11u101010 |
| 07 : | 10nuuuuuuuuuuuu11100n00un0u1001 | 11100001011111111111011 0000111100 |
| 08 : | 0001111100000000unnn11010001n001 | 10100010111100011011100 01u1001n1 |
| 09 : | 0000011111111111111110001n111un111 | u1001010000001111001100 10n1101u0 |
| 10 : | 11101101101111111 10100nu111uu011 | 00000010111100001010011 1111001011 |
| 11 : | 00111110010001010011011uu0n000u0 | 11110111011001001110101 01n1110n0 |
| 12 : | 01000110100011100011111 1nuu1u011 | 001101111111----------n111010 |
| 13 : | 101010000000----0--------01111u0 | 01010--------------------u0000u1 |
| 14 : | 00110001----------------1010010 | 10010------0----1--------00110n1 |
| 15 : | 10011--------------------10101n0 | n0110------0----1--------0111110 |
| 16 : | 0------------------------011000 | 10000--------------------11010u1 |
| 17 : | 1----------------------------n- | u1000------1----1--------u100111 |
| 18 : | 0------------------------------ | 01100------1----0--------01111u0 |
| 19 : | ------------------------------ | n1010--------------------1110100 |
| 20 : | ------------------------------ | u1000------1----------10000n1 |
| 21 : | ----------------------------n- | n1101--------------------u010011 |
| 22 : | ------------------------------ | 11101--------------------1100010 |
| 23 : | ----------------------------n- | u1011------0----------u110101 |
| 24 : | ------------------------------ | n1001--------------------0010110 |
| 25 : | ----------------------------u- | 00010--------------------n001011 |
| 26 : | ------------------------------ | u0001------1----------01110u0 |
| 27 : | ------------------------------ | 10111--------------------0011001 |
| 28 : | ------------------------------ | n1110--------------------1101001 |
| 29 : | ------------------------------ | u0000--------------------0010100 |
| 30 : | ------------------------------ | 01000--------------------0001001 |
| 31 : | ------------------------------ | 01011--------------------1000101 |
| 32 : | ------------------------------ | 00101--------------------1010111 |
| 33 : | ------------------------------ | 11000--------------------0010001 |
| 34 : | ------------------------------ | 01110--------------------00000n0 |
| 35 : | ----------------------------n- | 10101--------------------u101001 |
| 36 : | ------------------------------ | 10011--------------------10110u1 |
| 37 : | ------------------------------ | n1000--------------------01100u0 |
| 38 : | ----------------------------u- | n1001--------------------n010100 |
| 39 : | ------------------------------ | n0001--------------------11000n0 |
| 40 : | ------------------------------ | u0101--------------------1001001 |
| | ... | ... |

TAB. 6-5 – Chemin différentiel complet pour le second bloc, pas 1 à 40.

| $i$ | $\nabla_{A_i}$ | $\nabla_{W_i}$ |
|---|---|---|
| | ... | ... |
| 41 : | ---------- | n0100----------0010111 |
| 42 : | ---------- | u0000----------01100u1 |
| 43 : | ----------u- | 00111----------n101101 |
| 44 : | ---------- | 10001----------01011n0 |
| 45 : | ---------- | n0011----------1010000 |
| 46 : | ---------- | n0011----------1100111 |
| 47 : | ---------- | n0011----------0011000 |
| 48 : | ---------- | 11101----------10011u0 |
| 49 : | ----------u- | 01010----------n001000 |
| 50 : | ---------- | 01110----------11100n0 |
| 51 : | ---------- | n0111----------0111000 |
| 52 : | ---------- | n0001----------1101011 |
| 53 : | ---------- | n0100----------11100u0 |
| 54 : | ----------u- | 11000----------n000010 |
| 55 : | ---------- | 00111----------00001n0 |
| 56 : | ---------- | u1100----------10001u0 |
| 57 : | ----------u- | u0001----------n110000 |
| 58 : | ---------- | n1000----------1101011 |
| 59 : | ----------u- | u1111----------n0000u1 |
| 60 : | ----------u- | n0010----------n0100n0 |
| 61 : | ---------- | 01100----------10100n1 |
| 62 : | ---------- | 11001----------0101000 |
| 63 : | ---------- | 01100----------0000100 |
| 64 : | ---------- | n0011----------0101001 |
| 65 : | ---------- | 00101----------0101000 |
| 66 : | ---------- | 01011----------11101n0 |
| 67 : | ----------n- | 11111----------u100000 |
| 68 : | ---------- | 11110----------10100n1 |
| 69 : | ---------- | n0100----------1010011 |
| 70 : | ---------- | n0010----------00011n0 |
| 71 : | ----------n- | n0100----------u100001 |
| 72 : | ---------- | 10011----------10101u1 |
| 73 : | ---------- | n1001----------0010111 |
| 74 : | ---------- | n0101----------1101110 |
| 75 : | ---------- | u1111----------11001n1 |
| 76 : | ----------n- | 01100----------u111110 |
| 77 : | ---------- | 00001----------11010n0 |
| 78 : | ---------- | n0111----------1101000 |
| 79 : | ---------- | n0001----------0110011 |
| 80 : | ---------- | |

TAB. 6-6 – Chemin différentiel complet pour le second bloc, pas 41 à 80.

| | $1^{er}$ bloc | | $2^{nd}$ bloc | |
|---|---|---|---|---|
| | $M_1$ | $M'_1$ | $M_2$ | $M'_2$ |
| $W_0$ | 0x4643450b | 0x46434549 | 0x9a74cf70 | 0x9a74cf32 |
| $W_1$ | 0x41d35081 | 0x41d350c1 | 0x04f9957d | 0x04f9953d |
| $W_2$ | 0xfe16dd9b | 0xfe16dddb | 0xee26223d | 0xee26227d |
| $W_3$ | 0x3ba36244 | 0x3ba36204 | 0x9a06e4b5 | 0x9a06e4f5 |
| $W_4$ | 0xe6424055 | 0x66424017 | 0xb8408af6 | 0x38408ab4 |
| $W_5$ | 0x16ca44a0 | 0x96ca44a0 | 0xb8608612 | 0x38608612 |
| $W_6$ | 0x20f62444 | 0xa0f62404 | 0x8b7e0fea | 0x0b7e0faa |
| $W_7$ | 0x10f7465a | 0x10f7465a | 0xe17e363c | 0xe17e363c |
| $W_8$ | 0x5a711887 | 0x5a7118c5 | 0xa2f1b8e5 | 0xa2f1b8a7 |
| $W_9$ | 0x51479678 | 0xd147963a | 0xca079936 | 0x4a079974 |
| $W_{10}$ | 0x726a0718 | 0x726a0718 | 0x02f2a7cb | 0x02f2a7cb |
| $W_{11}$ | 0x703f5bfb | 0x703f5bb9 | 0xf724e838 | 0xf724e87a |
| $W_{12}$ | 0xb7d61841 | 0xb7d61801 | 0x37ffc03a | 0x37ffc07a |
| $W_{13}$ | 0xa5280003 | 0xa5280041 | 0x53aa8c43 | 0x53aa8c01 |
| $W_{14}$ | 0x6b08d26e | 0x6b08d26c | 0x90811819 | 0x9081181b |
| $W_{15}$ | 0x2e4df0d8 | 0xae4df0d8 | 0x312d423e | 0xb12d423e |

| Empreinte finale | | | | |
|---|---|---|---|---|
| $A_2$ | $B_2$ | $C_2$ | $D_2$ | $E_2$ |
| 0x6f84b892 | 0x1f9f2aae | 0x0dbab75c | 0x0afe56f5 | 0xa7974c90 |

TAB. 6-7 – Exemple d'une paire de messages pour une collision sur 2 blocs : $H(M_1, M_2) = H(M'_1, M'_2) = A_2||B_2||C_2||D_2||E_2$, obtenue conformément au chemin différentiel présenté dans les tableaux 6-3, 6-4, 6-5 et 6-6.

# 7

# Amélioration de la caractéristique linéaire

**Sommaire**



## 7-1 Introduction

Les résultats que nous présentons dans ce chapitre s'articulent autour de la proposition d'un nouvel algorithme de recherche de bons vecteurs de perturbations fondé sur un compromis différent des autres algorithmes présents dans la littérature. De l'utilisation de cet algorithme découle la première classification des vecteurs de perturbations utilisés pour les cryptanalyses de la fonction SHA-1. Cet algorithme a été proposé une première fois sous la forme d'un document ePrint [Man08]. Une version remaniée et étendue de l'article a été présentée lors de la conférence WCC'09 [Man09]. Afin d'évaluer les vecteurs de perturbations candidats, nous avons été amenés à construire des fonctions d'évaluations de ces vecteurs fondées strictement sur l'état de l'art de la littérature existante au moment de la rédaction du document ePrint. Cependant, ces évaluations ont été remises en cause ce qui nous a conduit à réaliser un certain nombre d'expériences dans le but de construire une fonction d'évaluation plus pertinente. Une partie des résultats obtenus lors de ces expériences fait l'objet d'une publication dans le journal *Designs Codes and Cryptography* [Man10].

Nous commencerons par introduire ce nouvel algorithme de recherche de vecteurs de perturbations. Nous décrirons son fonctionnement et détaillerons ce qui le distingue des autres algorithmes présents dans la littérature. Dans un second temps,

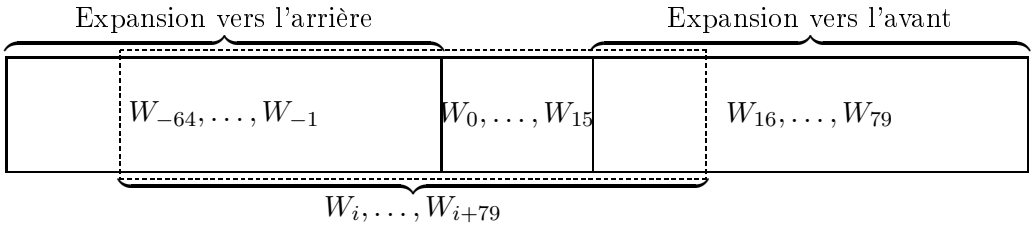



FIG. 7-1 – Message expansé étendu.

nous définirons la classification des vecteurs de perturbations que nous proposons. Puis, nous clorons ce chapitre par une revue des fonctions d'évaluation que nous avons utilisées lors de nos expérimentations et discuterons de leur adéquation avec la complexité d'une attaque par collision contre SHA-1.

## 7-2 Algorithme de recherche de vecteur de perturbations

### 7-2.1 Description de l'algorithme

L'algorithme que nous avons mis au point est principalement fondé sur une observation simple : la formule de récurrence utilisée par la fonction d'expansion de SHA-1 peut être définie selon deux directions. Nous nommons ces directions expansion vers l'avant et expansion vers l'arrière. On peut fixer les 16 mots de 32 bits $W_0, \ldots, W_{15}$ puis effectuer une expansion vers l'avant afin d'obtenir les mots $W_{16}, \ldots, W_{79}$ :

– Expansion vers l'avant :
$W_i = (W_{i-16} \oplus W_{i-14} \oplus W_{i-8} \oplus W_{i-3}) \lll 1$ pour $16 \leq i \leq 79$.

L'expansion vers l'avant constitue l'expansion standard telle que définie dans les spécifications de SHA-1. On peut aussi choisir de faire une expansion vers l'arrière qui permet d'obtenir les mots $W_{-64}, \ldots, W_{-1}$ définis par :

– Expansion vers l'arrière :
$W_i = (W_{i+16} \ggg 1) \oplus W_{i+13} \oplus W_{i+8} \oplus W_{i+2}$ pour $-64 \leq i \leq -1$.

N'importe quelle séquence de 80 mots de 32 bits consécutifs $W_i, \ldots, W_{i+79}$ avec $-64 \leq i \leq 0$ constitue un message expansé valide.

Nous qualifierons par le terme de "fenêtre d'information" les 16 mots de 32 bits $W_0, \ldots, W_{15}$. Pour une fenêtre d'information donnée, nous définissons l'ensemble des 144 mots de 32 bits obtenus en appliquant les deux expansions que nous venons de décrire comme étant le "message expansé étendu" défini par :

$$W_{-64}, \ldots, W_{-1}, W_0, \ldots, W_{15}, W_{16}, \ldots, W_{79},$$

Chaque message expansé étendu est composé de 65 messages expansé valides, chacun de ces messages valides constitue un candidat plausible comme vecteur de perturbation. Cette propriété constitue le coeur du principe de construction des vecteurs de perturbations candidats. La figure 7-1 illustre ce principe.

#### 7-2.1.1 Espace de recherche

La taille de l'espace comprenant tous les vecteurs de perturbations possibles est égale à $2^{16\times32}$, soit $2^{512}$. Une recherche exhaustive dans cet espace est donc hors

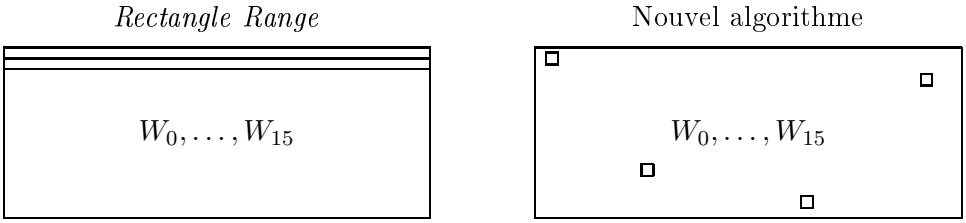



FIG. 7-2 – Dans le compromis du *Rectangle Range*, le poids de la fenêtre d'information est inférieur ou égal à 32, mais les perturbations ne sont autorisées que sur les positions 0 et 1. Dans le compromis de notre algorithme, le poids de la fenêtre d'information est limité à la valeur du paramètre $w$, mais les perturbations sont autorisées sur les positions $0, \ldots, 31$.

de portée. Les algorithmes de recherche de vecteur de perturbations mettent donc en oeuvre des heuristiques afin d'explorer un sous-espace particulier de l'espace de recherche.

Notre algorithme généralise le principe de l'algorithme décrit pour la première fois dans [WYY05d]. L'approche de Wang *et al.* consiste à choisir un espace de recherche sous la forme d'un domaine rectangulaire (*rectangle range*), défini de la façon suivante :

$$\{W_i = (0, \ldots, 0, w_{i,1}, w_{i,0}) | i = t, \ldots, t+15\} \text{ pour } t = 0, \ldots, 64.$$

Dans l'article de Wang *et al.*, les perturbations ne pouvaient être introduites qu'aux positions de bit 0 et 1. Le choix de ces deux positions étant dû au fait que la meilleure probabilité théorique de bon comportement est obtenue pour la position de bit 1. Cependant, Yajima *et al.* [YIN08, YIN09] utilisent une version étendue de cet algorithme autorisant l'insertion de perturbations sur la position de bit 31. Les tailles des sous-espaces de recherche de ces deux approches sont alors égales à $65 \times 2^{32}$ pour la version de l'algorithme utilisée par l'équipe de Wang et $65 \times 2^{48}$ pour l'équipe de Yajima. Le compromis choisi consiste donc à limiter les zones possibles pour l'insertion de perturbations, le nombre de perturbations insérées pouvant varier de 1 à 32 pour la version de Wang, de 1 à 48 pour la version de Yajima.

Notre approche considère un compromis différent illustré figure 7-2. Nous choisissons de limiter le nombre de perturbations susceptibles d'être introduites mais relâchons la contrainte de la position de bit où ces perturbations sont autorisées. En d'autres termes, nous limitons le poids de Hamming, noté $w$, de notre fenêtre d'information mais permettons aux perturbations d'apparaître n'importe où dans cette fenêtre. La taille de notre sous-espace de recherche est alors une fonction de $w$ égale à $65 \times \binom{w}{512}$. Le poids de Hamming de la fenêtre d'information constitue le premier paramètre utilisé par notre algorithme de recherche. L'utilisation d'un compromis différent permet à notre algorithme d'explorer des régions de l'espace de recherche qui ne sont couvertes ni par le sous-espace de Wang *et al.*, ni par le sous-espace étendu de Yajima *et al.*

#### 7-2.1.2 Heuristique d'évaluation

Afin de classer les vecteurs de perturbations candidats, il est nécessaire de définir une fonction de coût pour évaluer leur pertinence. Cette fonction de coût constitue le second paramètre utilisé par notre algorithme 4.

**Algorithm 4** Algorithme de recherche de vecteurs de perturbations

**Require:** $w > 0$, fonction de coût $f$
  **for all** fenêtre d'information de poids de Hamming inférieur ou égal à $w$ **do**
    générer le message expansé étendu correspondant
    **for all** 65 vecteurs de perturbations valides **do**
      évaluer l'efficacité avec la fonction de coût $f$
      stocker le vecteur présentant la meilleure évaluation
    **end for**
  **end for**
  **return** meilleur vecteur trouvé

Certains des algorithmes présents dans la littérature ont utilisé une fonction de coût fondée sur le poids de Hamming des 60 dernières coordonnées du vecteur de perturbations. C'est par exemple le cas de l'approche de Jutla and Patthak [JP05] qui ont aussi démontré que le poids de Hamming minimal calculé à partir des 60 dernières coordonnées d'un vecteur de perturbations est égal à 25. Leur fonction de coût est définie par le produit du poids de Hamming d'un vecteur candidat par une valeur égale à la moyenne des probabilités théoriques de bon comportement d'une collision locale. Paradoxalement, bien que cette estimation puisse paraître assez grossière, les vecteurs proposés dans leur article s'avèrent être particulièrement intéressants.

Une amélioration possible de cette approche consiste à évaluer plus finement le comportement de chaque collision locale. En effet, rappelons qu'une collision locale démarrant au pas 20 (ronde de $XOR$) possède une probabilité théorique de bon comportement égale à $2^{-2}$ si la perturbation initiale se situe sur la position de bit 1, $2^{-3}$ sur la position de bit 31 et $2^{-4}$ sur les positions restantes. En conséquence, un vecteur possédant un poids de Hamming supérieur peut finalement se révéler plus efficace qu'un vecteur avec un poids de Hamming minimal (égal à 25). Les autres heuristiques d'évaluations publiées tirent partie de cette analyse plus fine du comportement des collisions locales. Dans l'algorithme de Wang *et al.*, la probabilité de bon comportement de chaque collision locale est évaluée sous la forme d'un nombre de conditions à remplir (conformément à l'approche que nous avons qualifiée de déterministe). De plus, une heuristique particulière, la compression de bits "simple" est appliquée lorsque deux collisions locales sont adjacentes : ces collisions démarrent au même pas sur des positions de bit consécutives. Cette heuristique repose sur un fait vérifié expérimentalement : la probabilité de bon comportement de deux collisions locales consécutives est égale à la probabilité de bon comportement de la collision locale initiée sur la position de bit la plus petite dès lors que leurs directions sont opposées (condition que l'on peut garantir en imposant des contraintes sur les bits des mots de message expansés). Les heuristiques employées dans les articles [YIN08, YIN09] se fondent elles aussi sur un nombre de conditions à remplir, mais avec une analyse du comportement des collisions locales plus précise et une généralisation du principe de la compression de bits. Dans leur article de 2007 [DCM07], De Cannière *et*

*al.* précisent qu'ils utilisent leur propre algorithme pour rechercher de bons vecteurs de perturbations. Cependant, cet algorithme et l'heuristique d'évaluation associée ne sont détaillés dans aucun de leurs articles.

Pour les besoins de notre algorithme, nous avons été amenés à construire deux fonctions de coûts ad hoc. Ces fonctions seront explicitées dans la section 7-4.

### 7-2.2 Résultats expérimentaux

Nous avons tout d'abord choisi de rechercher des vecteurs de perturbations susceptibles d'être employées dans des attaques par collision sur deux blocs. Nous avons lancé l'exécution de notre algorithme en limitant successivement le poids de Hamming $w$ de la fenêtre d'information à 1, 2, 3 puis 4. Les temps d'exécution observés, sur un processeur Intel pentium 4 cadencé à 3,1 Ghz, sont de l'ordre d'une seconde pour $w = 1$, d'une dizaine de secondes pour $w = 2$, d'environ deux heures pour $w = 3$ et d'environ 700 heures pour $w = 4$.

Le traitement des résultats obtenus nous a amené à constater un certain nombre de faits expérimentaux :

- fait 1 : notre algorithme nous a permis de retrouver tous les vecteurs de perturbations publiés,
- fait 2 : tous les vecteurs de perturbations efficaces présentaient une certaine similarité dans leur profil,
- fait 3 : parmi les différents vecteurs présentant les meilleures performances, certains d'entre eux pouvaient être obtenus en effectuant une permutation circulaire des $W_i$ d'autres vecteurs,
- fait 4 : les perturbations de tous les vecteurs efficaces étaient principalement concentrées sur les bits les plus significatifs ou sur les bits les moins significatifs de leur mots $W_i$.

L'observation des faits 3 et 4 nous a conduit à ajouter une heuristique temporaire à notre algorithme consistant à ne pas autoriser de perturbations sur les positions de bits 5 à 25 ; l'ajout de cette contrainte permettant de réduire le temps d'exécution. Nous avons alors été à même d'étendre notre recherche de vecteurs jusqu'au paramètre $w = 6$. Cette nouvelle recherche a confirmé les faits que nous avions déjà observés, et aucun vecteur plus performant n'est apparu. Les meilleurs vecteurs de perturbations produits par notre algorithme ont tous été obtenus soit par une fenêtre d'information de poids exactement 1, soit par une fenêtre d'information de poids exactement 3. Le tableau 7-1 présente les deux fenêtres d'information conduisant aux meilleurs vecteurs de perturbations.

Nous avons aussi effectué une recherche de vecteurs de perturbations susceptibles de conduire à une collision sur un bloc, en imposant comme contrainte qu'aucune collision locale ne soit initiée après le pas 75. Nous avons pu alors expérimentalement vérifier que le vecteur le plus efficace pour une collision sur un bloc est le vecteur signalé pour la première fois par Wang *et al.* dans [WYY05b].

Nous avons testé de façon exhaustive tous les vecteurs pouvant être générés à partir d'une fenêtre d'information de poids de Hamming inférieur ou égal à 4. Nous avons de plus effectué des recherches partielles jusqu'au poids de Hamming 6. Cependant, cela ne représente qu'un "petit" sous-espace de l'ensemble de tous les vecteurs possibles, et de meilleurs vecteurs de perturbations pourraient exister. Malgré cela, il nous semble peu probable que des fenêtres d'information de poids supérieur à 6

| $i$ | Fenêtre d'information de poids 1 | Fenêtre d'information de poids 3 |
|---|---|---|
| 0 | -------------------------------- | -------------------------------- |
| 1 | -------------------------------- | o------------------------------- |
| 2 | -------------------------------- | -------------------------------- |
| 3 | -------------------------------- | o------------------------------- |
| 4 | -------------------------------- | -------------------------------- |
| 5 | -------------------------------- | -------------------------------- |
| 6 | -------------------------------- | -------------------------------- |
| 7 | -------------------------------- | -------------------------------- |
| 8 | -------------------------------- | -------------------------------- |
| 9 | -------------------------------- | -------------------------------- |
| 10 | -------------------------------- | -------------------------------- |
| 11 | -------------------------------- | -------------------------------- |
| 12 | -------------------------------- | -------------------------------- |
| 13 | -------------------------------- | -------------------------------- |
| 14 | -------------------------------- | -------------------------------- |
| 15 | --------------------------------o | --------------------------------o |

TAB. 7-1 – Fenêtres d'information conduisant aux vecteurs de perturbations les plus efficaces.

puissent conduire à l'obtention de meilleurs candidats.

Il est intéressant de remarquer que certaines des observations que nous avons réalisées avaient été évoquées dans la littérature. Wang *et al.* [WYY05d] ont énoncé que différents choix de position de bit produisaient des vecteurs de perturbations constituants des copies d'autres vecteurs avec le même poids de Hamming. Rijmen et Oswald [RO05] ont noté que les vecteurs obtenus grâce à leur algorithme, désignés sous le terme de mots de codes dans leur article, possédaient un grand nombre de mots $W_i$ en commun. Jutla et Patthak [JP05] ont indiqué que le premier mot de code proposé dans leur article avait déjà été signalé par Wang *et al.* De la même façon, Pramstaller *et al.* [PRR05] ont souligné que leur vecteur était le même que celui utilisé par Wang *et al.* à une permutation des indices prêt. Cependant, aucune publication antérieure à notre article [Man08] n'expliquait, ni ne proposait un modèle capable de prendre en compte ces observations. L'interprétation des faits expérimentaux que nous avons observés nous a conduit à conclure que les vecteurs de perturbations efficaces pouvaient être classifiés. Nous présentons cette classification dans la section suivante.

## 7-3 Classification des vecteurs de perturbations

### 7-3.1 Relation d'équivalence

Nous avons choisi de modéliser et de générer les vecteurs de perturbations à l'aide d'une fenêtre d'information et de ce que nous avons nommé les messages expansés étendus. Sous cette forme, il est facile de voir que permuter circulairement les $W_i$ d'une fenêtre d'information produira des vecteurs qui seront des versions permutées

les uns des autres. De la même façon, les 65 vecteurs de perturbations composant un message expansé étendu ne diffèrent qu'à partir de l'indice $i$ ($-64 \leq i \leq 0$) d'où les messages expansés valides débutent. En se fondant sur ces deux propriétés, nous pouvons définir une relation d'équivalence.

**Définition 7.1 (Équivalence de deux vecteurs de perturbations)**
*Soient le vecteur de perturbations* $V =< V_0, V_1, \ldots, V_{79} >$ *et le vecteur de perturbations* $W =< W_0, W_1, \ldots, W_{79} >$. *Les vecteurs* $V$ *et* $W$ *sont équivalents si au moins une des deux propriétés suivantes est vérifiée :*

- *il existe un entier* $j$, $0 \leq j < 32$, *tel que :* $W_i = V_i \lll j$ *pour* $0 \leq i \leq 79$ *(on peut obtenir le premier vecteur en effectuant une permutation circulaire de tous les mots de 32 bits du second vecteur), ou*
- *les deux vecteurs appartiennent au même message expansé étendu.*

Cette relation d'équivalence permet de classifier l'ensemble de tous les vecteurs en fonction de la fenêtre d'information à partir de laquelle ils peuvent être générés. A notre connaissance, il s'agit de la première classification proposée pour les vecteurs de perturbations utilisés lors des attaques par collision contre les fonctions SHA-0 et SHA-1. L'avantage principal de définir une telle classification réside dans le fait qu'elle permet d'expliquer et de modéliser les différentes remarques et observations qui ont été faites dans la littérature.

Une fois cette relation d'équivalence définie, nous avons constaté expérimentalement que les vecteurs de perturbations les plus performants résident dans seulement deux classes d'équivalence. Ces classes, baptisées classe de type-I et classe de type-II, correspondent aux deux fenêtres d'information que nous avons explicitées dans le tableau 7-1. Il est intéressant de remarquer que tous les vecteurs de perturbations publiés sont des vecteurs de type-I ou de type-II. La classe de type-I (fenêtre d'information de poids 1 dans le tableau 7-1) contient les vecteurs équivalents au vecteur publié pour la première fois par Wang *et al.* dans [WYY05b]. La classe de type-II (fenêtre d'information de poids 3 dans le tableau 7-1) regroupe les vecteurs équivalents au vecteur publié pour la première fois par Jutla et Patthak dans [JP05]. Le tableau 7-2 ainsi que le tableau 7-3 présentent de façon synthétique l'ensemble des vecteurs publiés à ce jour et montrent qu'ils sont tous de type-I ou de type-II. Dans ces deux tableaux, la notation $\ggg i$ indique que pour retrouver le vecteur publié dans la référence, il suffit d'effectuer une permutation circulaire vers la gauche de $i$ bits de chacun des 80 mots de 32 bits).

### 7-3.2 Nouvelle Notation

Nous introduisons à présent une nouvelle notation pour les vecteurs de perturbations : nous noterons I(i, j) (respectivement II(i, j)) les vecteurs de type-I (respectivement type-II) pouvant être générés de la façon suivante :

- effectuer une permutation circulaire de $j$ bits vers la gauche des 16 mots de 32 bits de la fenêtre d'information du type-I (respectivement du type-II),
- effectuer une expansion vers l'arrière $i$ fois,
- effectuer une expansion vers l'avant $64 - i$ fois.

Le tableau 7-4 donne la notation correspondante aux vecteurs de perturbations connus. Cette notation fournit un moyen de représenter de façon compacte les vecteurs de perturbations appartenant à ces deux classes.

| Vecteur<br><br>Type-I | Wang *et al.* [WYY05c] [WYY05d] [SKP07] [JP07]<br><br>$\ggg 2$ | Rijmen & Oswald [RO05]<br><br>$\ggg 1$ | | | Jutla & Patthak [JP05]<br><br>*Code word*1 | <br><br>*Code word*3 | Pramstaller *et al.* [PRR05]<br><br>$\ggg 2$ | De Cannière & Rechberger [DCR06]<br><br>$\ggg 2$ |
|---|---|---|---|---|---|---|---|---|
| ---------- | | | | | 0 | | | |
| -oo------- | | | | | 1 | 0 | | |
| ---------- | | | | | 2 | 1 | 0 | |
| -o-o------ | 0 | | | | 3 | 2 | 1 | |
| o--------- | 1 | | | | 4 | 3 | 2 | |
| o--------- | 2 | | | | 5 | 4 | 3 | |
| o-o------- | 3 | | | | 6 | 5 | 4 | |
| -o-------- | 4 | 0 | | | 7 | 6 | 5 | |
| ---------- | 5 | 1 | | | 8 | 7 | 6 | |
| -oo------- | 6 | 2 | | | 9 | 8 | 7 | |
| o--------- | 7 | 3 | | | 10 | 9 | 8 | |
| o--------- | 8 | 4 | 0 | | 11 | 10 | 9 | |
| o--------- | 9 | 5 | 1 | | 12 | 11 | 10 | |
| ---------- | 10 | 6 | 2 | 0 | 13 | 12 | 11 | |
| ---------- | 11 | 7 | 3 | 1 | 14 | 13 | 12 | |
| -o-------- | 12 | 8 | 4 | 2 | 15 | 14 | 13 | |
| ---------- | 13 | 9 | 5 | 3 | 16 | 15 | 14 | |
| o-o------- | 14 | 10 | 6 | 4 | 17 | 16 | 15 | 0 |
| o--------- | 15 | 11 | 7 | 5 | 18 | 17 | 16 | 1 |
| o-o------- | 16 | 12 | 8 | 6 | 19 | 18 | 17 | 2 |
| ---------- | 17 | 13 | 9 | 7 | 20 | 19 | 18 | 3 |
| o--------- | 18 | 14 | 10 | 8 | 21 | 20 | 19 | 4 |
| ---------- | 19 | 15 | 11 | 9 | 22 | 21 | 20 | 5 |
| oo-------- | 20 | 16 | 12 | 10 | 23 | 22 | 21 | 6 |
| ---------- | 21 | 17 | 13 | 11 | 24 | 23 | 22 | 7 |
| o--------- | 22 | 18 | 14 | 12 | 25 | 24 | 23 | 8 |
| o--------- | 23 | 19 | 15 | 13 | 26 | 25 | 24 | 9 |
| -o-------- | 24 | 20 | 16 | 14 | 27 | 26 | 25 | 10 |
| ---------- | 25 | 21 | 17 | 15 | 28 | 27 | 26 | 11 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| ---------- | 61 | 57 | 53 | 51 | 64 | 63 | 62 | 47 |
| ---------- | 62 | 58 | 54 | 52 | 65 | 64 | 63 | 48 |
| ---------- | 63 | 59 | 55 | 53 | 66 | 65 | 64 | 49 |
| ----------o | 64 | 60 | 56 | 54 | 67 | 66 | 65 | 50 |
| ---------- | 65 | 61 | 57 | 55 | 68 | 67 | 66 | 51 |
| ---------- | 66 | 62 | 58 | 56 | 69 | 68 | 67 | 52 |
| ----------o- | 67 | 63 | 59 | 57 | 70 | 69 | 68 | 53 |
| ---------- | 68 | 64 | 60 | 58 | 71 | 70 | 69 | 54 |
| ---------- | 69 | 65 | 61 | 59 | 72 | 71 | 70 | 55 |
| ----------o-- | 70 | 66 | 62 | 60 | 73 | 72 | 71 | 56 |
| ---------- | 71 | 67 | 63 | 61 | 74 | 73 | 72 | 57 |
| ----------o- | 72 | 68 | 64 | 62 | 75 | 74 | 73 | 58 |
| ----------o--- | 73 | 69 | 65 | 63 | 76 | 75 | 74 | 59 |
| ---------- | 74 | 70 | 66 | 64 | 77 | 76 | 75 | 60 |
| ---------- | 75 | 71 | 67 | 65 | 78 | 77 | 76 | 61 |
| ----------o----- | 76 | 72 | 68 | 66 | 79 | 78 | 77 | 62 |
| ---------- | 77 | 73 | 69 | 67 | | 79 | 78 | 63 |
| ----------o-o- | 78 | 74 | 70 | 68 | | | 79 | 64 |
| ----------o----- | 79 | 75 | 71 | 69 | | | | 65 |
| ----------oo- | | 76 | 72 | 70 | | | | 66 |
| ---------- | | 77 | 73 | 71 | | | | 67 |
| ----------o------ | | 78 | 74 | 72 | | | | 68 |
| ----------o--- | | 79 | 75 | 73 | | | | 69 |
| ----------o-o--- | | | 76 | 74 | | | | 70 |
| ----------o------ | | | 77 | 75 | | | | 71 |
| ----------o--- | | | 78 | 76 | | | | 72 |
| ---------- | | | 79 | 77 | | | | 73 |
| ----------o----o--- | | | | 78 | | | | 74 |
| ---------- | | | | 79 | | | | 75 |
| ----------o-o----- | | | | | | | | 76 |
| ----------o--------- | | | | | | | | 77 |
| ----------oo--o-- | | | | | | | | 78 |
| ---------- | | | | | | | | 79 |

TAB. 7-2 – Vecteurs de perturbations publiés de type-I. La notation $\ggg i$ indique que pour retrouver le vecteur de la référence, il suffit d'effectuer une permutation circulaire vers la gauche de $i$ bits de chacun des 80 mots correspondants.

| Vecteur<br>Type-II | Yajima *et al.* [YIN08]<br>$\ggg 2$ | De Cannière *et al.* [DCM07]<br>$\ggg 2$ | Jutla & Patthak [JP05]<br>*Codeword*2 |
|---|---|---|---|
| oo-o---------- | 0 | | |
| o---------- | 1 | | |
| ooo---------- | 2 | | |
| o-o---------- | 3 | | |
| ---o---------- | 4 | | |
| o---------- | 5 | | 0 |
| ooo---------- | 6 | | 1 |
| --o---------- | 7 | | 2 |
| oo-o---------- | 8 | | 3 |
| o---------- | 9 | | 4 |
| ooo---------- | 10 | | 5 |
| o-o---------- | 11 | 0 | 6 |
| ---o---------- | 12 | 1 | 7 |
| o---------- | 13 | 2 | 8 |
| ooo---------- | 14 | 3 | 9 |
| --o---------- | 15 | 4 | 10 |
| oo---------- | 16 | 5 | 11 |
| o---------- | 17 | 6 | 12 |
| -oo---------- | 18 | 7 | 13 |
| o---------- | 19 | 8 | 14 |
| oo---------- | 20 | 9 | 15 |
| o---------- | 21 | 10 | 16 |
| o-o---------- | 22 | 11 | 17 |
| o---------- | 23 | 12 | 18 |
| ooo---------- | 24 | 13 | 19 |
| ---------- | 25 | 14 | 20 |
| ⋮ | ⋮ | ⋮ | ⋮ |
| ---------- | 61 | 51 | 57 |
| ---------- | 62 | 52 | 58 |
| ---------- | 63 | 53 | 59 |
| ---------- | 64 | 54 | 60 |
| ---------- | 65 | 55 | 61 |
| ---------- | 66 | 56 | 62 |
| ---------- | 67 | 57 | 63 |
| ---------- | 68 | 58 | 64 |
| ---------- | 69 | 59 | 65 |
| ---------- | 70 | 60 | 66 |
| ----------o | 71 | 61 | 67 |
| ---------- | 72 | 62 | 68 |
| ---------- | 73 | 63 | 69 |
| ----------o- | 74 | 64 | 70 |
| ----------o | 75 | 65 | 71 |
| ---------- | 76 | 66 | 72 |
| ----------o-- | 77 | 67 | 73 |
| ----------o- | 78 | 68 | 74 |
| ----------o- | 79 | 69 | 75 |
| ----------o--- | | 70 | 76 |
| ----------o-- | | 71 | 77 |
| ---------- | | 72 | 78 |
| ----------o--o- | | 73 | 79 |
| ----------o--- | | 74 | |
| ----------o-o- | | 75 | |
| ----------o----- | | 76 | |
| ----------o-oo- | | 77 | |
| ---------- | | 78 | |
| ----------o--o-o | | 79 | |

TAB. 7-3 – Vecteurs de perturbations publiés de type-II. La notation $\ggg i$ indique que pour retrouver le vecteur de la référence, il suffit d'effectuer une permutation circulaire vers la gauche de $i$ bits de chacun des 80 mots correspondants.

| Vecteurs de perturbations | Notation |
|---|---|
| Wang *et al.* CRYPTO 2005 [WYY05c] | |
| 58 pas | I(43, 2) |
| 80 pas | I(49, 2) |
| Rijmen & Oswald CT-RSA 2005 [RO05] | |
| *Codeword*1 | I(45, 1) |
| *Codeword*2 | I(41, 1) |
| *Codeword*3 | I(39, 1) |
| Jutla & Patthak ePrint 2005 [JP05] | |
| *Codeword*1 | I(52, 0) |
| *Codeword*2 | II(52, 0) |
| *Codeword*3 | I(51, 0) |
| Pramstaller *et al.* IMA 2005 [PRR05] | I(50, 2) |
| De Cannière & Rechberger ASIACRYPT 2006 [DCR06] | I(35, 2) |
| De Cannière *et al.* SAC 2007 [DCM07] | II(46, 2) |
| Yajima *et al.* ASIACCS 2008 [YIN08] | II(56, 2) |

TAB. 7-4 – Nouvelle notation des vecteurs de perturbations publiés.

Bien que des vecteurs de perturbations d'une même classe soient par définition équivalents, l'efficacité de ces vecteurs peut varier dans de grandes proportions. Nous décrivons les fonctions de coût que nous avons définies pour réaliser l'évaluation de l'efficacité des vecteurs de perturbations dans la section suivante.

## 7-4 Fonction d'évaluation

Dans le but de comparer l'efficacité de différents vecteurs de perturbations, notre algorithme prend comme paramètre une fonction de coût. Une telle fonction se doit, pour un vecteur de perturbations donné, de refléter la complexité d'une attaque par collision fondée sur ce vecteur. Cependant, la complexité de ces attaques est largement subordonnée à différents facteurs, et notamment aux techniques d'accélération de recherche mises en oeuvre lors de l'attaque. De plus, l'expérience montre que les vecteurs de perturbations influent sur le comportement des générateurs automatiques de caractéristiques non-linéaires. Dès lors, définir une fonction de coût optimale semble être un objectif particulièrement difficile.

Dans le cadre d'une attaque pratique de recherche de collision, nous préconisons d'utiliser notre algorithme dans le cadre d'une approche en deux étapes. La première étape consiste à définir une ou plusieurs fonctions de coût fournissant une première évaluation brute de l'efficacité d'un vecteur, afin d'effectuer un balayage "rapide" de l'espace de recherche et d'obtenir un ensemble de bons candidats potentiels. La seconde étape consiste à évaluer, en prenant en compte tous les facteurs inhérents à cette attaque, les vecteurs de cet ensemble pour choisir le plus performant d'entre eux. C'est l'approche que nous avons mis en oeuvre avec Thomas Peyrin afin de produire une collision pour la fonction SHA-0 [MP08].

Nous allons à présent, décrire les deux fonctions de coût que nous avons utilisées lors de nos expérimentations.

### 7-4.1 Fonctions de coût

Nous avons choisi de rechercher des vecteurs de perturbations susceptibles d'être utilisés pour construire une collision sur deux blocs pour la fonction SHA-1 (80 pas). Il est cependant tout à fait possible d'utiliser notre algorithme pour rechercher des vecteurs de perturbations pour des versions réduites. Il suffit pour cela de définir une fonction de coût en adéquation avec le nombre de pas voulu.

#### 7-4.1.1 Rationale

Pour construire nos fonctions de coût, nous nous sommes appuyés sur un certain nombre d'hypothèses présentes dans la littérature.

- Nous avons choisi de commencer l'évaluation des vecteurs au pas 21. En effet, les articles de Wang *et al.* [WYY05b, WYY05c] ont montré qu'il était possible de trouver des modifications de messages avancées permettant de s'assurer du bon comportement des collisions locales au moins jusqu'à ce pas.
- De la même façon que dans [DCM07], nous avons choisi d'ignorer les conditions liées aux retenues pour les pas 79 et 80. En effet pour le premier bloc de la collision, ces conditions peuvent être gérées par la caractéristique non-linéaire appliquée au second bloc. De plus notre objectif consistant à obtenir une estimation de l'efficacité d'un vecteur de perturbation donné, il ne nous a pas semblé complètement déraisonnable de ne pas prendre en compte ces conditions.
- Nos deux fonctions de coût intègrent aussi les techniques de compression de bits, décrites dans [WYY05b] et [YIN08].

L'évaluation des vecteurs de perturbations candidats que nous avons choisie repose sur l'approche probabiliste. La fonction de coût 1 utilise les probabilités théoriques de bon comportement des collisions locales que l'on peut retrouver dans l'article de Mendel *et al.* [MPR06]. La fonction de coût 2 utilise les probabilités données par les tables B.1 et B.2 de l'annexe B de la thèse de Thomas Peyrin [Pey08].

Nous rappelons que ces fonctions n'ont pas pour but de donner une évaluation précise et définitive de la complexité d'une attaque par collision contre SHA-1. Leur objectif est de permettre de déterminer de façon grossière mais rapide quels sont parmi tous les vecteurs candidats, les vecteurs susceptibles d'être les plus performants. En outre, nous soulignons le fait que la fonction de coût ne constitue qu'un paramètre de l'algorithme et que celui-ci est à même de fonctionner avec n'importe quelle fonction de coût.

#### 7-4.1.2 Évaluation des vecteurs publiés

Nous avons réuni table 7-5 les différentes évaluations, obtenues au moyen de nos deux fonctions de coût, correspondant aux vecteurs de perturbations présents dans la littérature. Il est intéressant de remarquer que les évaluations obtenues sont très proches des complexités revendiquées lors des attaques par collisions menées contre des versions réduites de SHA-1 [WYY05b, DCR06, DCM07]. Ce qui semble montrer que bien que nos fonctions de coût soient relativement grossières, elles reflètent de

| Vecteurs de perturbations | Complexité revendiquée $\log_2$ | Évaluation 1 $-\log_2$ | Évaluation 2 $-\log_2$ |
|---|---|---|---|
| Wang *et al.* CRYPTO 2005 [WYY05c] | | | |
| 58 pas | 33 | 35 | 35 |
| 80 pas | 69 | 73 | 73 |
| Rijmen & Oswald CT-RSA 2005 [RO05] | | | |
| 80 pas | | | |
| *Codeword*1 | - | 90 | 90 |
| *Codeword*2 | - | 97 | 97 |
| *Codeword*3 | - | 102 | 102 |
| Jutla & Patthak ePrint 2005 [JP05] | | | |
| 80 pas | | | |
| *Codeword*1 | - | 70 | 76 |
| *Codeword*2 | - | 65 | 69 |
| *Codeword*3 | - | 71 | 76 |
| Pramstaller *et al.* IMA 2005 [PRR05] | | | |
| 80 pas | - | 73 | 73 |
| De Cannière & Rechberger ASIACRYPT 2006 [DCR06] | | | |
| 64 pas | 35 | 34 | 34 |
| 80 pas | - | 94 | 94 |
| De Cannière *et al.* SAC 2007 [DCM07] | | | |
| 70 pas | 44 | 43 | 43 |
| 80 pas | - | 88 | 88 |
| Yajima *et al.* ASIACCS 2008 [YIN08] | | | |
| 80 pas | 70 (72) CVCs | 75 | 75 |

TAB. 7-5 – Comparaison de l'évaluation, relativement aux fonctions de coûts que nous avons choisies, de l'efficacité des vecteurs de perturbations publiés. L'évaluation 1 (respectivement 2) se fonde sur l'article de Mendel *et al.* (respectivement la thèse de Peyrin).

façon assez pertinente l'efficacité des vecteurs de perturbations. Nous notons aussi que le vecteur de perturbations le plus performant, relativement aux fonctions de coûts que nous avons choisies, est le vecteur désigné sous le nom de *Codeword*2 décrit pour la première fois par Jutla et Patthak dans [JP05].

#### 7-4.1.3 Autre fonction

Les deux fonctions de coût que nous avons employées sont des fonctions conservatrices. Dans le papier ePrint [Man08] que nous avons proposé, nous avions défini et utilisé une fonction plus optimiste dont l'évaluation des vecteurs commence au pas 25. En effet, dans [WYY05b], les auteurs exhibent des modifications de message avancées permettant de commencer l'évaluation de leur vecteur à partir de ce pas. Cependant, ces modifications de message ne sont pas indépendantes du vecteur de perturbations : elles doivent être construites spécifiquement. Par conséquent, il n'y a pas de certitude pour un vecteur de perturbations donné sur l'existence de modifi-

cations de message avancées efficaces. Cependant si l'on accepte l'hypothèse de leur existence, le meilleur vecteur de perturbations obtenu avec cette fonction optimiste est le vecteur noté $\mathrm{II}(49, 0)$. McDonald *et al.* ont présenté durant la Rump session d'EUROCRYPT'09 une attaque par collision fondée sur ce vecteur.

### 7-4.2 De l'efficacité à la complexité

Une fonction de coût est destinée à évaluer l'efficacité d'un vecteur de perturbations. Considérons à présent le lien entre cette évaluation de l'efficacité et la complexité d'une attaque par collision contre SHA-1 fondée sur ce vecteur.

On peut trouver dans la littérature principalement trois approches utilisées pour procéder à une évaluation de la complexité des attaques par collision :

- le décompte de conditions [WYY05d, WYY05b, WYY05c, SKP07, MP08, YIN08, YIN09],
- le calcul de probabilités [CJ98, BC04, BCJ05, MPR06],
- l'estimation du facteur de travail [DCR06, DCM07, JP07].

Pour un vecteur de perturbations donné la complexité des attaques est évaluée à partir d'un certain pas, soit en additionnant le nombre de conditions à remplir, soit en multipliant les probabilités de bon comportement de chaque collision locale correspondant aux perturbations définies par ce vecteur. L'estimation du facteur de travail se fonde, quant à elle, sur le nombre moyen de noeuds d'un arbre devant être visités afin de trouver une paire de messages qui vérifie une certaine caractéristique différentielle. À ces évaluations peuvent s'ajouter un certain nombre de paramètres qui dépendent essentiellement des techniques d'accélération de recherche employées. De plus, des analyses plus fines du comportement des collisions locales peuvent, elles aussi, influencer l'évaluation de la complexité.

Les fonctions de coût que nous avons choisi de définir reposent sur un calcul de probabilités démarrant au pas 21, l'hypothèse sous-jacente étant que l'on peut s'assurer du bon comportement des collisions locales du pas 16 au pas 20 indépendamment du vecteur de perturbations. Il s'agit d'une hypothèse discutable, mais relativement commune dans la littérature. De plus, comme nous l'avons déjà souligné auparavant, le processus de génération de caractéristiques non-linéaires ainsi que des éventuelles techniques d'accélération de recherche peuvent présenter des comportements différents en fonction du vecteur de perturbations considéré. Aussi, nous ne pouvons pas revendiquer qu'il est possible de monter simplement et directement une attaque par collision contre la fonction SHA-1 à partir du meilleur vecteur trouvé par notre algorithme.

Cependant, cela ne dégrade pas la valeur de l'algorithme que nous proposons car il peut être utilisé avec quelque fonction de coût définie pour refléter au mieux les contraintes spécifiques liée à l'attaque.

# 8

# Évaluation Statistique du comportement des collisions locales

**Sommaire**



## 8-1 Introduction

Afin d'évaluer la complexité d'une attaque par collision, l'hypothèse suivante est généralement admise : les probabilités de bon comportement des collisions locales sont indépendantes. Selon cette hypothèse, la probabilité de bon comportement d'un enchevêtrement de collisions locales est considérée comme étant le produit des probabilités de bon comportement des collisions locales impliquées. Cette hypothèse est aussi appliquée, sous une forme équivalente, pour le décompte de conditions. Cependant, dans l'article de Chabaud et Joux [CJ98] un comportement particulier, ce que nous avons dénommé cas pathologique, est identifié concernant la fonction $IF$. Deux collisions locales consécutives dans les 15 premiers pas de la fonction de mise à jour des registres conduit à un chemin différentiel impossible, qui nécessite l'emploi d'une caractéristique non-linéaire. Thomas Peyrin décrit dans sa thèse [Pey08] un comportement similaire susceptible d'être observé pour la fonction $MAJ$. La solution consiste alors à s'assurer que les directions de deux collisions locales apparaissant entre les pas 36 et 55 soient opposées. En outre, la technique de compression de bit utilisée dans [WYY05b, YIN08, YIN09] permet d'augmenter les probabilités de bon comportement de collisions locales adjacentes en choisissant avec précaution les directions des différentes collisions locales impliquées. Ces exemples montrent que

l'hypothèse d'indépendance des collisions locales ne se vérifie pas dans un certain nombres de cas de figure. C'est à partir de se constat, que nous avons défini une procédure expérimentale et mené un certain nombre d'expérimentations visant à confronter cette hypothèse d'indépendance. Les résultats présentés dans ce chapitre seront publiés dans la prochaine édition du journal Design, Codes and Cryptography [Man10].

## 8-2 Procédure expérimentale

Dans le but de mesurer les probabilités de bon comportement d'enchevêtrements de collisions locales nous avons défini une procédure expérimentale présentée dans l'algorithme 5. Cette procédure requiert quatre paramètres différents utilisés pour décrire l'information nécessaire pour construire les paires de messages candidates et les traiter. Le premier paramètre $S$ correspond au nombre de pas qu'il est nécessaire d'effectuer pour "arriver au bout" de l'enchevêtrement de collisions locales. Le deuxième paramètre définit ce que nous nommons le patron de collision locale. Le patron correspond à la description complète de l'enchevêtrement de collision locales :

- le nombre de collisions locales,
- la position à laquelle elles sont insérées (la première collision, en terme de position de bit, est toujours insérée à la position de bit 0),
- le numéro du bit où est insérée chaque collision,
- la direction de chaque collision.

En ce qui concerne la direction d'une collision locale, nous utiliserons pour la définir la notion de signe positif ou négatif. Une collision locale insérée sur une position de bit $b$ correspond à la différentielle :

- $(+2^b, -2^{b+5}, -2^b, -2^{b-2}, -2^{b-2}, -2^{b-2})$ pour une collision de signe positif
- $(-2^b, +2^{b+5}, +2^b, +2^{b-2}, +2^{b-2}, +2^{b-2})$ pour une collision de signe négatif.

Le troisième paramètre spécifie la fonction de ronde utilisée et le dernier paramètre impose le nombre de test à effectuer pour l'expérience.

Par exemple, considérons le cas de deux collisions locales consécutives sur les positions de bits 1 et 4, de signe positif et se déroulant toutes deux dans une ronde de $MAJ$. Le nombre de pas $S$ est fixé à 7 : au pas 7, les séquences de corrections auront étés intégralement appliquées et une collision des registres devrait être observée. Le paramètre $f$ correspondant à la fonction de ronde est défini comme étant $MAJ$. Le patron de collision locale pour cet enchevêtrement particulier est alors :

$$(+2^1, -2^6 + 2^4, -2^1 - 2^9, -2^{31} - 2^4, -2^{31} - 2^2, -2^{31} - 2^2, -2^2).$$

Le nombre de test $T$ est fixé à $2^{20}$, ce qui permet d'obtenir une précision relative de 0.01 pour le logarithme de la probabilité.

La procédure basique d'expérimentation consiste à tirer aléatoirement un état initial pour les registres et à générer une paire de message aléatoire se conformant au patron de collision locale. Puis, chaque message est utilisé pour calculer parallèlement les nouveaux états des registres pour le nombre de pas $S$ nécessaire. Finalement, les deux états finals sont comparés afin de vérifier si on retrouve effectivement une collision. Cette expérience est reproduite le nombre de fois $T$ nécessaire pour obtenir la précision désirée. Pour un type de patron donné toutes les positions de bits possibles sont testées.

**Algorithm 5** Mesure des probabilités de bon comportement de patrons de collisions locales

**Require:** nombre de pas $S$, patron de collision locale, fonction de ronde $f$, nombre de tests $T$
  **for** $b = 0$ to 31 **do**
    **for** $i = 1$ to 3 **do**
      $m_i \leftarrow 0$
      **for** $t = 1$ to $T$ **do**
        Tirer aléatoirement un état initial $(A_0, A_{-1}, A_{-2}, A_{-3}, A_{-4})$
        Tirer aléatoirement $S$ valeurs de 32 bits et appliquer le patron de collision locale pour obtenir la paire de message $W_0, \ldots, W_{S-1}$ et $W'_0, \ldots, W'_{S-1}$
        Calculer $(A_1, \ldots, A_S)$ et $(A'_1, \ldots, A'_S)$ selon la fonction de mise à jour des registres avec la fonction de ronde $f$
        **if** $(A_{S-4} \oplus A'_{S-4} = A_{S-3} \oplus A'_{S-3} = A_{S-2} \oplus A'_{S-2} = A_{S-1} \oplus A'_{S-1} = A_S \oplus A'_S = 0)$ **then**
          $m_i \leftarrow m_i + 1$
        **end if**
      **end for**
    **end for**
    $P_b \leftarrow (m_1 + m_2 + m_3)/(3 \times T)$
  **end for**
  **return** $P_0, \ldots, P_{31}$

Une fois cette procédure définie, nous avons testé le comportement de différents patrons de collisions locales présents dans les vecteurs de perturbations de type-I et de type-II.

## 8-3 Résultats

### 8-3.1 Collision locale isolée

Le premier patron de collision locale que nous avons considéré est le cas d'une collision locale isolée. Les différentes probabilités de bon comportement que nous avons mesurées sont listées dans le tableau 8-1. Les valeurs correspondant aux probabilités théoriques de bon comportement sont représentées entre parenthèses. La notation [4.85,4.87] indique que les probabilités mesurées varient entre les valeurs $2^{-4.85}$ et $2^{-4.87}$ selon les différentes positions de bit considérées. Ces résultats sont compatibles avec l'analyse fondée sur l'impact des retenues proposée dans [MPR06]. Nous signalons cependant une erreur[3] contenue dans l'article concernant la probabilité de bon comportement initiée à la position de bit 31 dans une ronde de $MAJ$. Cette probabilité est égale à $2^{-4}$ et non $2^{-3}$.

Nous pouvons observer que les valeurs mesurées sont égales aux probabilités théoriques pour les positions de bit 1, 26 et 31. Les probabilités obtenues pour les autres positions sont légèrement meilleures que les probabilité théoriques du fait de l'effet de propagation des retenues. Cependant, cet effet peut être détruit s'il est amené à dépasser la position de bit 31, car à cette position la retenue est ignorée

[3] Cette erreur a aussi été remarquée par Thomas Peyrin dans sa thèse.

(addition modulo $2^{32}$). Les perturbations initiées sur les positions de bit 1, 26 ou 31 correspondent, ou impliquent des corrections relatives à des modifications sur la position de bit 31. Ceci est le fait des valeurs des permutations circulaires présentes dans la formule de mise à jour des registres. Les collisions locales initiées sur ses positions de bit ne peuvent donc pas bénéficier de l'effet de propagation des retenues qui améliore les probabilités sur les autres positions de bit. Pour la fonction $XOR$ la probabilité de bon comportement en fonction de la position de bit $b$, notée $p(XOR, b)$ est égale à :

$$p(XOR,b) = \begin{cases} 2^{-4} + 2^{-6} & \text{for } b = 0 \\ 2^{-2} & \text{for } b = 1 \\ \sum_{k=1}^{27-b} 2^{-4k} & \text{for } b = 2, \ldots, 26 \\ 2.2^{-4.(32-b)} + \sum_{k=1}^{31-b} 2^{-4k} & \text{for } b = 27, \ldots, 31 \end{cases}$$

Pour la fonction $MAJ$, cette probabilité est égale à :

$$p(MAJ,b) = \begin{cases} 2^{-4} + 2^{-8} & \text{for } b = 0 \\ 2^{-4} & \text{for } b = 1 \\ \sum_{k=1}^{27-b} 2^{-4k} & \text{for } b = 2, \ldots, 26 \\ \sum_{k=1}^{32-b} 2^{-4k} & \text{for } b = 27, \ldots, 31 \end{cases}$$

| Patron | ----------o |
|---|---|

| Position de bit $b$ | | 0 | 1 | 2, ..., 25 | 26 | 27,28,29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|
| Fonction de ronde | Signes | Probabilités mesurées $-\log_2$ | | | | | | |
| IF | + | 4.87 | 5.00 | [4.85, 4.87] | 5.00 | 4.86 | 4.83 | 4.01 |
| | − | 4.87 | 5.00 | [4.85, 4.87] | 5.00 | 4.86 | 4.84 | 4.00 |
| | | (5) | (5) | (5) | (5) | (5) | (5) | (4) |
| XOR | + | 3.68 | 2.00 | [3.90, 3.92] | 4.00 | [3.90, 3.91] | 3.83 | 3.00 |
| | − | 3.68 | 2.00 | [3.90, 3.92] | 4.00 | 3.91 | 3.83 | 3.00 |
| | | (4) | (2) | (4) | (4) | (4) | (4) | (3) |
| MAJ | + | 3.91 | 4.00 | [3.90, 3.92] | 4.00 | [3.90, 3.91] | 3.92 | 4.00 |
| | − | 3.92 | 4.00 | [3.90, 3.92] | 4.00 | 3.91 | 3.91 | 4.00 |
| | | (4) | (4) | (4) | (4) | (4) | (4) | (4) |

TAB. 8-1 – Probabilités de bon comportement mesurées pour une collision locale isolée se déroulant entièrement au sein d'une même fonction de ronde. Les valeurs présentées sont les moyennes observées sur 3 expériences comprenant $2^{20}$ tests chacune.

Nous avons présenté dans la table 8-1 les probabilités de bon comportement pour la fonction de ronde $IF$. Dans nos expériences suivantes, nous avons choisi d'ignorer cette fonction car on attend de la caractéristique non-linéaire de prendre en main le comportement des collisions locales pour cette ronde.

### 8-3.2 Collision locales adjacentes

Nous avons mesuré les probabilités de bon comportement relatives à des collisions locales adjacentes. Ces expérimentations prennent en compte l'ensemble des cas correspondant à la compression de bit simple décrite dans l'article de Wang *et al.* [WYY05b]. Les probabilités mesurées sont détaillées dans le tableau 8-2. Nous remarquons que le cas de la fonction de ronde $XOR$ a été analysé de façon théorique par Mendel *et al.* dans [MPR06]. Les résultats que nous obtenons confirment leur analyse. Le comportement de la fonction $MAJ$ est similaire à celui de la fonction $XOR$, et le principe de la compression de bit se vérifie bien.

Nous rappelons que la compression de bits consiste à forcer l'application de l'effet de propagation de la retenue (*Carry Effect*) en spécifiant de façon délibérée les signes des perturbations. Cet effet repose sur une simple propriété des puissances de 2 :

$$+2^i = -2^i - 2^{i+1} - 2^{i+2} - \cdots - 2^{i+(j-1)} + 2^{i+j}.$$

Une série de perturbations adjacentes peut donc être vue comme une unique perturbation positionnée à la même position de bit que la première des perturbations impliquées, dès lors que les directions de ces perturbations sont choisies de façon adéquate. La spécification de ces signes peut se faire de façon simple en influant directement sur les bits des mots de message expansés, moyennant la perte d'un certain nombre de degrés de liberté. Du point de vue de la propagation à l'état interne, cette collision locale compressée se comporte effectivement comme une seule collision locale. En effet l'égalité arithmétique garantit qu'avec une probabilité $1/2$ (la même valeur que la probabilité pour qu'il n'apparaisse pas de retenue dans dans le cas d'une collision locale unique), la perturbation propagée au registre se bornera à la position de bit de la première perturbation. Malencontreusement, la compression de bit n'est possible que lorsque la propagation de la retenue n'est pas arrêté par l'addition modulo $2^{32}$. Combiné aux deux permutations circulaires présentes dans la fonction de mise à jour des registres, cela implique qu'il ne peut y avoir de compression de bit sur les positions 1 et 2, 26 et 27 ou 31 et 0.

Nous avons mené des expérimentations supplémentaires afin de vérifier le fonctionnement du principe de compression de bit à un nombre de collisions locales adjacentes supérieur à deux : les résultats observés sont concluants.

### 8-3.3 Collision locales consécutives

Le cas des collisions locales consécutives présente un intérêt particulier car il correspond aux cas pathologiques déjà identifiés dans la littérature. L'étude de ce type d'enchevêtrement de collisions locales nous a permis d'une part de montrer que le cas pathologique identifié pour la fonction $MAJ$ n'est constitué que pour un nombre restreint de positions de bit, et d'autre part de mettre à jour de nouveaux cas pathologiques.

Le tableau 8-3 liste les résultats obtenus pour deux collisions locales positionnées sur le même numéro de bit et démarrant lors de deux pas immédiatement consécutifs. Nous rappelons que les seuls résultats publiés précédemment pour ce patron particulier, concernent les cas pathologiques relatifs aux fonctions de ronde $IF$ et $MAJ$. Pour la fonction $MAJ$, la solution proposée dans la littérature pour éviter un chemin différentiel impossible consiste à imposer aux deux collisions locales des signes opposés.

| Patron | ----------oo | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Position de bit $b$ | 0 | 1 | 2, ..., 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 |
| Fonction de ronde | Signes | Probabilités mesurées $-\log_2$ | | | | | | | | |
| XOR | $--$ | 6.00 | 5.91 | [6.90, 6.91] | 6.96 | 8.00 | 7.90 | 6.91 | 6.87 | 6.36 | 7.00 |
| | $++$ | 6.00 | 5.91 | [6.90, 6.90] | 6.96 | 8.00 | 7.91 | 6.90 | 6.87 | 6.36 | 7.00 |
| | | (6) | (6) | (8) | (8) | (8) | (8) | (8) | (8) | (8) | (7) |
| | $-+$ | 3.68 | 5.91 | [3.90, 3.91] | 3.91 | 3.91 | 7.90 | 3.91 | 3.91 | 3.90 | 3.83 |
| | $+-$ | 3.68 | 5.90 | [3.90, 3.91] | 3.91 | 3.91 | 7.91 | 3.90 | 3.91 | 3.90 | 3.83 |
| | | (4) | (6) | (4) | (4) | (4) | (8) | (4) | (4) | (4) | (4) |
| MAJ | $--$ | 8.00 | 7.90 | [6.90, 6.91] | 6.96 | 7.99 | 7.90 | 6.91 | 6.91 | 6.95 | 8.00 |
| | $++$ | 8.00 | 7.91 | [6.90, 6.91] | 6.96 | 8.00 | 7.91 | 6.91 | 6.91 | 6.95 | 8.00 |
| | | (8) | (8) | (8) | (8) | (8) | (8) | (8) | (8) | (8) | (8) |
| | $-+$ | 3.91 | 7.91 | [3.90, 3.91] | 3.91 | 3.91 | 7.91 | 3.91 | 3.91 | 3.91 | 3.91 |
| | $+-$ | 3.91 | 7.91 | [3.90, 3.91] | 3.91 | 3.91 | 7.91 | 3.91 | 3.91 | 3.91 | 3.91 |
| | | (4) | (8) | (4) | (4) | (4) | (8) | (4) | (4) | (4) | (4) |

TAB. 8-2 – Probabilités de bon comportement de deux collisions locales adjacentes se déroulant entièrement au sein d'une même fonction de ronde. Les valeurs présentées sont les moyennes observées sur 3 expériences comprenant $2^{24}$ tests chacune.

Nos expériences montrent que même lorsque les signes sont égaux, l'existence de chemins différentiels impossibles ne semble être établie que pour les positions de bit 1, 26 et 31. Pour les autres positions de bit, certaines paires de message sont parvenues à former une collision, avec cependant une probabilité de bon comportement inférieure (proche de $2^{-10}$) à la probabilité théorique de bon comportement avec signes opposés ($2^{-8}$). Lorsque les signes des collisions locales sont effectivement opposés, on peut observer que les probabilités mesurées sont meilleures que les probabilités théoriques (autour de $2^{-6}$ au lieu de $2^{-8}$).

Cette amélioration des probabilités théoriques est aussi présente pour la fonction de ronde $XOR$. À l'exception de la position de bit 1, et quelle que soit la combinaison de signes, les probabilités mesurées sont meilleures que prévues. Cela est particulièrement intéressant pour la position de bit 31. Pour ce type d'enchevêtrement de collisions locales, on peut remarquer que la position de bit 1 n'est plus exclusivement celle qui donne la meilleure probabilité de bon comportement.

### 8-3.4 Collisions locales alternées

Le premier type de patron présentant un enchevêtrement de collision locales alternées que nous avons testé est constitué de deux collisions locales insérées à la même position de bit et commençant aux pas $i$ et $i+2$. Les résultats obtenus sont rassemblés dans le tableau 8-4.

Pour la fonction $XOR$, le comportement observé est conforme à ce que nous attendions. Cependant pour la fonction $MAJ$, les probabilités mesurées sont nettement inférieures aux probabilités théoriques. Il n'y a correspondance que pour des perturbations de même direction insérées à la position de bit 1, et pour des perturbations de directions opposées insérées à la position de bit 31. Pour la position de bit 1 avec

| Patron | ----------o<br>----------o | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Position de bit $b$ | 0 | 1 | 2, ..., 24 | 25 | 26 | 27,28,29 | 30 | 31 |
| Fonction de ronde | Signes | Probabilités mesurées $-\log_2$ | | | | | | |
| XOR | $--$ | 5.91 | 4.00 | [5.97, 5.98] | 5.98 | 6.00 | [5.97, 5.98] | 5.91 | 4.00 |
| | $-+$ | 5.91 | 4.00 | [5.97, 5.98] | 5.98 | 6.00 | 5.98 | 5.91 | 4.00 |
| | $+-$ | 5.91 | 4.00 | 5.98 | 5.98 | 6.00 | 5.98 | 5.91 | 4.00 |
| | $++$ | 5.91 | 4.00 | [5.97, 5.98] | 5.98 | 6.00 | [5.97, 5.98] | 5.91 | 4.00 |
| | | (8) | (4) | (8) | (8) | (8) | (8) | (8) | (6) |
| MAJ | $--$ | 10.01 | $> 24$ | [9.88, 9.92] | 9.99 | $> 24$ | [9.91, 9.92] | 9.99 | $> 24$ |
| | $++$ | 10.00 | $> 24$ | [9.88, 9.92] | 10.00 | $> 24$ | [9.90, 9.91] | 10.01 | $> 24$ |
| | | $(\infty)$ | $(\infty)$ | $(\infty)$ | $(\infty)$ | $(\infty)$ | $(\infty)$ | $(\infty)$ | $(\infty)$ |
| | $-+$ | 5.98 | 6.00 | [5.97, 5.98] | 5.98 | 6.00 | 5.98 | 5.98 | 6.00 |
| | $+-$ | 5.98 | 6.00 | [5.97, 5.98] | 5.98 | 6.00 | 5.98 | 5.98 | 6.00 |
| | | (8) | (8) | (8) | (8) | (8) | (8) | (8) | (8) |

TAB. 8-3 – Probabilités de bon comportement de deux collisions locales consécutives (démarrant au pas $i$ et $i+1$) se déroulant entièrement au sein d'une même fonction de ronde. Les valeurs présentées sont les moyennes observées sur 3 expériences comprenant $2^{24}$ tests chacune. La notation "$> 24$" désigne le fait qu'aucune collision n'a été trouvée lors des trois expériences, la probabilité de bon comportement correspondante est donc strictement inférieure à $2^{-24}$. La notation $\infty$ désigne un chemin différentiel théoriquement impossible.

des directions opposées, et la position de bit 31 avec une même direction, aucune collision ne s'est jamais produite au cours des 3 expériences comportant $2^{24}$ essais chacune. D'un point de vue statistique, cela peut signifier que la probabilité d'obtenir une collision est inférieure à $2^{-25}$, ou bien que nous sommes confrontés à des chemins différentiels impossibles et donc à de nouveaux cas pathologiques. Le même type de comportement peut être observé pour la position de bit 26, et ce quelque soit la configuration des directions des collisions locales. Nous en déduisons qu'il s'agit là d'un réel nouveau cas pathologique. L'observation de ces comportements n'a, à notre connaissance, jamais fait l'objet d'une publication antérieure à notre article [Man10].

Nous remarquons de plus, que la probabilité théorique n'est atteinte que dans seulement 4 configurations :

- pour la position de bit 1 lorsque les directions sont identiques,
- pour la position de bit 31 lorsque les directions sont opposées.

Ailleurs, la probabilité mesurée est proche de $2^{-12}$, quand la probabilité théorique attendue est $2^{-8}$.

Nous avons conduit d'autres expérimentations pour 3 et 4 collisions locales alternées. Bien que nous ne présentions pas ces résultats dans cette thèse, nous avons pu observer de nombreuses autres anomalies pour ces patrons.

| Patron | ----------o<br>----------<br>----------o | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Position de bit $b$ | 0 | 1 | 2, ..., 24 | 25 | 26 | 27,28,29 | 30 | 31 |
| Fonction de ronde | Signes | Probabilités mesurées $-\log_2$ | | | | | | |
| XOR | $--$ | 7.36 | 4.00 | [7.81, 7.82] | 7.82 | 8.00 | [7.81, 7.82] | 7.66 | 6.00 |
| | $-+$ | 7.36 | 4.00 | [7.79, 7.80] | 7.83 | 8.00 | [7.77, 7.81] | 7.66 | 6.00 |
| | $+-$ | 7.35 | 4.00 | [7.79, 7.81] | 7.82 | 8.00 | [7.78, 7.80] | 7.66 | 6.00 |
| | $++$ | 7.36 | 4.00 | [7.81, 7.82] | 7.82 | 8.00 | [7.81, 7.82] | 7.66 | 6.00 |
| | | (8) | (4) | (8) | (8) | (8) | (8) | (8) | (6) |
| MAJ | $--$ | 11.88 | 8.00 | [11.77, 11.82] | 11.91 | $> 24$ | [11.80, 11.82] | 11.91 | $> 24$ |
| | $++$ | 11.91 | 7.99 | [11.79, 11.82] | 11.90 | $> 24$ | [11.81, 11.82] | 11.91 | $> 24$ |
| | $+-$ | 11.89 | $> 24$ | [11.58, 11.62] | 11.90 | $> 24$ | [11.60, 11.63] | 11.93 | 8.00 |
| | $-+$ | 11.92 | $> 24$ | [11.58, 11.63] | 11.90 | $> 24$ | [11.58, 11.63] | 11.89 | 8.00 |
| | | (8) | (8) | (8) | (8) | (8) | (8) | (8) | (8) |

TAB. 8-4 – Probabilités de bon comportement de deux collisions locales alternées (démarrant au pas $i$ et $i+2$) se déroulant entièrement au sein d'une même fonction de ronde. Les valeurs présentées sont les moyennes observées sur 3 expériences comprenant $2^{24}$ tests chacune. La notation "$> 24$" désigne le fait qu'aucune collision n'a été trouvée lors des trois expériences, la probabilité de bon comportement correspondante est donc strictement inférieure à $2^{-24}$.

## 8-4 Conclusion

L'ensemble des vecteurs de perturbations utilisés dans les cryptanalyses qui ont abouti à l'obtention de collisions locales pour les fonctions SHA-0 et des versions réduites de la fonction SHA-1 concentrent leurs perturbations sur la position de bit 1. Ce choix délibéré des cryptanalystes se fonde sur les probabilités théoriques de bon comportement (ou les nombres de conditions) héritées des premières études menées pour ces fonctions. Les expériences que nous avons conduites nous montrent que ce choix ne peut plus être considéré comme le seul choix possible. De fait la position de bit 31 peut, dans certain cas, se révéler tout aussi intéressante lorsque des collisions locales sont enchevêtrées. De plus, nous avions remarqué au chapitre 5 section 5-3.4, que l'instanciation des directions des collisions locales se fait généralement (à l'exception du cas pathologique de la fonction $MAJ$) de façon uniforme. Bien que le choix de cette instanciation ne fasse l'objet d'aucun commentaire dans ces cryptananlyses, les résultats que nous avons observés indiquent qu'il semble nécessaire de le justifier au vu des comportements qui apparaissent lors d'instanciations différentes. Nos expériences ont aussi permis de mettre à jour au moins un nouveau cas pathologique pour la fonction $MAJ$ et de montrer que le cas pathologique rapporté pour la première fois par Chabaud et Joux ne constitue pas toujours un chemin différentiel impossible.

Finalement nous pensons, que l'hypothèse assez commune dans la littérature, qui consiste à considérer les collisions locales comme indépendantes demande à être étudiée de façon approfondie. Nous ne sommes pas en mesure, à l'heure où nous

rédigeons ce document, de présenter une étude capable d'expliquer les comportements que nous avons remarqués. Cependant, nous espérons pouvoir poursuivre nos recherches dans cette direction.

# Troisième partie

# Conception de nouvelles fonctions

# 9

# Les fonctions XOR-Hash et FSB

**Sommaire**



## 9-1 La fonction XOR-Hash

Nous décrivons dans ce chapitre une construction de fonction de hachage cryptographique itérative, munie d'une transformation de sortie et fondée sur une fonction de compression construite à partir d'une fonction à sens unique. La construction que nous proposons est de type "coquille vide" (*Empty Shell Construction*). L'objectif de ce type de construction est de fournir un mode opératoire utilisant une ou plusieurs briques de base pour lesquelles un certain nombre de propriétés sont exigées. Sous l'hypothèse que ces briques possèdent effectivement les propriétés requises, la construction garantit un certain niveau de sécurité relativement aux attaques considérées.

La fonction XOR-Hash est le fruit d'une collaboration avec Nicolas Sendrier. L'article que nous avons rédigé a fait l'objet d'une présentation publique en 2007 au *Western European Workshop on Research in Cryptology*, organisé à Bochum en Allemagne [MS07].

XOR-Hash repose sur l'utilisation d'une nouvelle fonction de compression fondée sur une fonction à sens unique $f : \{0,1\}^r \to \{0,1\}^r$. La fonction $f$ est appliquée à la variable de chaînage et à un nombre maximum de $t-1$ blocs de message. Dès lors que $t$ valeurs ont été traitées, elles sont combinées au moyen d'un xor bit à bit afin de former la variable de chaînage suivante. Nous utilisons l'algorithme de Merkle-Damgård comme extenseur de domaine et appliquons une transformation de sortie finale $g$.

En limitant le nombre $t$ de valeurs combinées, nous faisons reposer la sécurité de notre construction sur des problèmes difficiles de théorie des codes et plus précisément sur la difficulté de décodage d'un code linéaire aléatoire. Pour les paramètres proposés, nous considérons que la meilleure attaque de décodage est l'algorithme du paradoxe des anniversaires généralisé de Wagner [Wag02]. Selon cette conjecture, nous obtenons une sécurité de $r/(2 + \log_2 t)$ bits contre les attaques par collisions. Afin d'obtenir une fonction de hachage de taille $n$ bits avec $n/2$ bits de sécurité, nous utilisons une transformation de sortie finale avec $r$ bits d'entrée et une sortie de $n = 2r/(2 + \log_2 t)$ bits.

### 9-1.1 Travaux connexes

Nous présentons dans cette section, les différents travaux qui nous ont conduit à proposer cette nouvelle construction.

**Famille de fonctions FSB.** En 2005 [AFS05], Augot *et al.* ont présenté une famille de fonctions de compression fondées sur la théorie des codes. Ces fonctions utilisent une fonction de syndrome régulier, consistant à "xorer" un petit nombre $t$ des colonnes d'une matrice aléatoire donnée comme paramètre. Les résistances aux collisions et au calcul d'antécédent de cette famille de fonctions de compression se fondent sur une réduction vers des problèmes difficiles de théorie des codes. Les auteurs ont démontré que le décodage de syndrome régulier ainsi que le décodage de syndrome régulier double sont des problèmes NP-complets. Ces fonctions de compression sont paramétrables et leurs paramètres sont déterminés respectivement aux deux meilleures attaques connues qui sont l'algorithme du paradoxe des anniversaires généralisé de Wagner [Wag02] et l'algorithme de décodage par ensemble d'information de Canteaut-Chabaud [CC98].

La fonction de hachage cryptographique FSB repose sur une instance de cette famille de fonctions de compression, nous décrirons la fonction FSB dans le chapitre suivant.

**XOR-MAC.** En 1995 [BGR95], Bellare *et al.* ont proposé de nouvelles méthodes pour construire des codes d'authentification de message appelés schémas XOR (*XOR-schemes*). Ces schémas reposent sur l'utilisation d'une fonction pseudo-aléatoire comme primitive. Une telle fonction peut être construite à partir d'un schéma de chiffrement par bloc idéal ou à partir de la fonction de compression d'une fonction de hachage cryptographique idéale. Le calcul du code d'authentification d'un message au moyen de l'algorithme XOR-MAC se déroule en trois étapes :

1. découpage du message en une collection de blocs,
2. application de la fonction pseudo-aléatoire à chacun des blocs, pour obtenir une collection d'images pseudo-aléatoires,
3. xor bit à bit de l'ensemble de ces images pour obtenir le MAC.

Bellare *et al.* ont prouvé que pour une famille sûre de fonctions pseudo-aléatoires, les schémas XOR-MAC fondés sur cette famille sont sûrs.

**Le paradigme de randomisation puis combinaison.** En 1997 [BM97], Bellare et Micciancio ont introduit un nouveau paradigme pour la conception de fonctions de hachage résistantes aux collisions. Le paradigme dit de randomisation puis combinaison (*Randomize-then-Combine Paradigm*) est une application aux fonctions de hachage des méthodes employées pour XOR-MAC.

Le message $x$ est vu comme une séquence de blocs $x = x_1 \dots x_\ell$. Chaque bloc $x_i$ est traité via la fonction de "randomisation" $f$ pour produire une valeur aléatoire $y_i = f(\langle i \rangle \parallel x_i)$, où la notation $\langle i \rangle$ désigne la représentation binaire du bloc d'indice $i$ et la notation $\parallel$ l'opération de concaténation. Puis, les valeurs aléatoires sont combinées au moyen de l'opérateur de "combinaison" $\odot$. La valeur finale hachée est alors $h(x) = y_1 \odot y_2 \odot \cdots \odot y_\ell$. Une illustration de ce paradigme se trouve Figure 9-1.

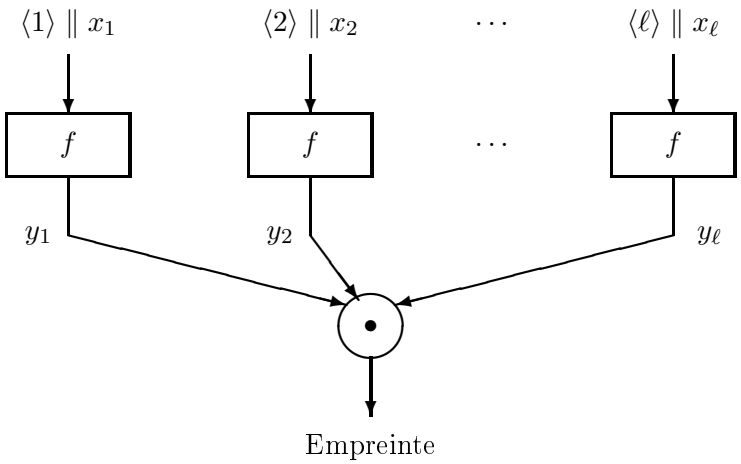



FIG. 9-1 – Paradigme de randomisation puis combinaison. Le calcul de l'empreinte correspondant à un message $x = x_1 \dots x_\ell$ se déroule en deux étapes : les blocs de message sont randomisés au moyen de la fonction $f$ et du codage de leur indice puis, les valeurs obtenues sont combinées pour produire l'empreinte.

L'avantage principal de ce type de construction réside dans son caractère incrémental. Si l'empreinte d'un message $x$ modifié en $x'$ a déjà été calculée, la nouvelle empreinte correspondant au message $x'$ peut être obtenue de façon efficace. Plutôt que de recalculer entièrement cette empreinte, on peut mettre à jour l'empreinte précédente en un temps proportionnel au montant des modifications apportées à $x$ pour obtenir $x'$.

Bellare et Micciancio ont proposé différentes solutions fondées sur différents opérateurs de combinaisons, trois de ces solutions possèdent des réductions de sécurité. Par exemple, la résistance aux collisions de la fonction MuHASH, qui utilise une multiplication comme opérateur de combinaison, peut se réduire à la résolution du problème du logarithme discret sous l'hypothèse que la fonction de randomisation $f$ est modélisée en tant que fonction de hachage idéale. L'opérateur de combinaison xor n'a pas été retenu par Bellare et Micciancio. En effet, la variante XHASH s'est révélée ne pas être sûre même si la fonction $f$ est un oracle aléatoire. Il existe une attaque utilisant l'algèbre linéaire efficace pour des messages très longs.

Cependant, nous soutenons qu'une fonction de hachage fondée sur le xor comme

opérateur de combinaison peut être sûre, tant que le nombre de combinaisons est inférieur ou égal à un paramètre de sécurité. La fonction XOR-Hash est fondée sur ce paradigme.

**Rumba20.** En 2007 [Ber07a], une nouvelle construction utilisant le principe de [BM97] a été proposée. Bernstein a présenté lors du Workshop ECRYPT sur les fonctions de hachage cryptographiques la fonction Rumba20. Cette fonction de compression est fondée sur le paradigme de randomisation puis combinaison avec un nombre de combinaisons fixé à 4 xor bit à bit.

La randomisation est faite à l'aide de 4 différentes instances $f_1, f_2, f_3$ et $f_4$ du schéma de chiffrement à flot Salsa20 [Ber07b]. Chaque instance $f_i$ est une expansion de 384 bits vers 512 bits, $f_i : \{0,1\}^{384} \rightarrow \{0,1\}^{512}$. L'entrée de la fonction de compression (la variable de chaînage précédente et le bloc de message courant) est découpée en 4 sous-blocs $(x_1, x_2, x_3, x_4)$. La sortie de la fonction de compression est égale à $f_1(x_1) \oplus f_2(x_2) \oplus f_3(x_3) \oplus f_4(x_4)$.

L'algorithme du paradoxe des anniversaires généralisé de Wagner est considéré comme étant l'attaque la plus efficace contre cette fonction. Les arguments de sécurité de Rumba20 reposent sur une estimation détaillée, décrite dans l'article, du coût de cette attaque.

## 9-1.2 Description de XOR-Hash

Nous présentons tout d'abord dans cette section, les considérations de conception que nous avons prises en compte pour construire notre schéma. Nous décrivons ensuite la fonction XOR-Hash selon un point de vue algorithmique pour finalement discuter de son efficacité en terme de taux de hachage.

### 9-1.2.1 Conception

Afin d'obtenir une fonction à la fois sûre et efficace, la conception de notre proposition repose sur plusieurs considérations que nous décrivons ici.

**Paradigme de randomisation puis combinaison.** Les auteurs de l'article de 1997, se sont principalement intéressés aux opérateurs de combinaison. En effet, Bellare et Micciancio préconisent d'utiliser une fonction de hachage standard comme fonction de randomisation, et basent leur analyse de sécurité sur le fait de modéliser cette fonction comme un oracle aléatoire.

Tout comme pour l'algorithme XOR-MAC, la fonction que nous proposons utilise une fonction à sens-unique $f : \{0,1\}^r \rightarrow \{0,1\}^r$ pour la randomisation et le xor bit à bit pour la combinaison.

**Construction *hybride*.** La construction hybride est une approche intermédiaire entre le paradigme de randomisation puis combinaison et les constructions de fonctions de compression classiques. Le résultat de chaque combinaison est stocké dans un accumulateur $H$, pré-rempli avec la randomisation d'une valeur initiale $IV$. Le paramètre de sécurité $t$ constitue le nombre maximum de valeurs randomisées qui seront combinées. L'idée fondamentale de la construction hybride consiste à nourrir l'accumulateur $H$ jusqu'à ce que le paramètre de sécurité $t$ soit atteint. Lorsque $t-1$

combinaisons ont été faites, la valeur courante de l'accumulateur $H$ est tronquée et utilisée comme nouvelle valeur initiale $\tilde{H}$. Pour des messages de taille multiple de $t-1$, le comportement de la construction hybride est similaire à une construction itérative de type Merkle-Damgård.

**Modèle d'attaques.** Pour les fonctions de la famille FSB et la fonction Rumba20, on considère que la meilleure attaque connue est l'algorithme du paradoxe des anniversaires généralisé de Wagner. Le principe de construction de XOR-Hash est similaire, bien que différent, à ces deux propositions. Il semble donc raisonnable de considérer cette attaque comme la plus significative lors de l'évaluation de la sécurité de XOR-Hash. Nous discuterons des impératifs de sécurité relatifs à la fonction de randomisation $f$ et à la fonction de transformation finale $g$ dans la section 9-1.3.

**Transformation de sortie.** Au vu des analyses de sécurité décrites dans [AFS05] et [Ber07a], on peut déduire que dans le but de résister à l'attaque du paradoxe des anniversaires généralisé, la taille de la variable de chaînage doit être de l'ordre de quelques centaines de bits. Dans le cadre de XOR-Hash, nous avons décidé d'utiliser une transformation de sortie afin de réduire la taille de l'empreinte. Cette transformation de sortie $g : \{0,1\}^r \rightarrow \{0,1\}^n$ est une fonction de compression et donc $n < r$.

#### 9-1.2.2 Construction

Nous détaillons la fonction XOR-Hash selon un point de vue algorithmique décrit dans l'algorithme 6. Une illustration graphique reprenant cette description est aussi proposée par la figure 9-2.

Soit $m$ le message à hacher, nous supposons qu'il inclut un rembourrage standard. Nous découpons ce message rembourré en $\ell$ blocs, $m = m_1 m_2 \dots m_\ell$, où chaque bloc $m_i$ est de taille $r - \lceil \log_2 t \rceil$. L'algorithme prend comme argument une variable de chaînage initiale $IV$ de taille $r - \lceil \log_2 t \rceil$ ainsi que les $\ell$ blocs $m_i$. De façon similaire à l'algorithme XOR-MAC, et dans le but de se prémunir d'une attaque par permutation de blocs, nous préfixons chaque valeur destinée à être randomisée par la fonction $f$ par une représentation binaire de son indice $\langle j \rangle$ pour $0 \leq j < t$. L'algorithme se déroule alors de la façon suivante :

1. La première étape consiste à calculer la valeur initiale devant être stockée dans l'accumulateur : $H = f(\langle 0 \rangle \parallel IV)$.
2. Puis deux compteurs sont initialisés : le compteur $i$ désigne le numéro du bloc de message courant et le compteur $j$ le nombre de valeurs randomisées destinées à être combinées.
3. Pour chaque bloc de message :
   (a) Si $t$ valeurs ont déjà été combinées : tronquer les $r - \lceil \log_2 t \rceil$ bits les plus à droite de $H$, les utiliser comme variable de chaînage $\tilde{H}$ afin de calculer la nouvelle valeur initiale de l'accumulateur puis recommencer un cycle de combinaisons en posant $j = 1$.
   (b) Combiner la valeur randomisée courante $f(\langle j \rangle \parallel m_i)$ avec la valeur courante de l'accumulateur, et stocker le résultat.

(c) Incrémenter les compteurs $i$ et $j$.

4. Lorsque tous les blocs de message ont été traités, appliquer la transformation de sortie $g$ à la valeur contenue par l'accumulateur $H$.

La sortie de l'algorithme $g(H)$ constitue l'empreinte du message.

**Algorithm 6** XOR-Hash

**Require:** $IV, m_1, \ldots, m_\ell \in \{0,1\}^{r-\lceil \log_2 t \rceil}$
  $H \leftarrow f(\langle 0 \rangle \parallel IV)$
  $i \leftarrow 1\,;\, j \leftarrow 1$
  **while** $i \leq \ell$ **do**
    **if** $j = t$ **then**
      $\tilde{H} \leftarrow \mathtt{chop}_{r-\lceil \log_2 t \rceil}(H)$
      $H \leftarrow f(\langle 0 \rangle \parallel \tilde{H})$
      $j \leftarrow 1$
    **end if**
    $H \leftarrow H \oplus f(\langle j \rangle \parallel m_i)$
    $i \leftarrow i+1\,;\, j \leftarrow j+1$
  **end while**
  **return** $g(H)$

Il est important de noter que nous n'instantions pas les fonctions $f$ et $g$. Le principe de notre construction de fonction de hachage munie d'une transformation de sortie consiste d'une part à proposer une nouvelle fonction de compression fondée sur une fonction à sens unique, et d'autre part à évaluer la sécurité de notre construction relativement aux propriétés de cette fonction de compression et de la transformation de sortie. Nous discuterons dans la section 9-1.3 des impératifs de sécurité que nous avons identifié pour ces deux fonctions.

#### 9-1.2.3 Évaluation de l'efficacité

**Taux de hachage.** La notion de taux de hachage a été introduite en 1993 par Preneel [Pre93], pour les fonctions de hachage fondées sur des schémas de chiffrement par bloc. Dans l'article [KL94], Knudsen et Lai ont défini le taux de hachage comme le nombre de blocs de message de taille $m$ bits traités sur le nombre d'appels, en chiffrement ou déchiffrement, à un schéma de chiffrement par bloc de taille $m$ bits. Cette définition constitue la mesure normalisée pour évaluer l'efficacité des fonctions de hachage fondées sur des schémas de chiffrement par bloc.

Bien qu'il soit possible de construire une fonction à sens unique au moyen d'un schéma de chiffrement par bloc, dans notre construction, nous n'exigeons pas formellement cette propriété de la fonction $f$. Cependant, nous définissons le taux de hachage relativement à la fonction à sens-unique $f$, comme le nombre de bits traités par $f$ sur le nombre de bits traités par la fonction XOR-Hash que nous notons ici $h$. Le taux de hachage de notre construction est alors :

$$\mathrm{Hr}_f = \frac{\text{bits traités par } f}{\text{bits de message traités par } h} = \frac{tr}{(t-1)(r - \lceil \log_2 t \rceil)}.$$

Nous remarquons que du fait de notre construction hybride $\mathrm{Hr}_f$ peut varier. Par exemple, pour des messages de longueur inférieure ou égale à $r - \lceil \log_2 t \rceil$, $\mathrm{Hr}_f$ est

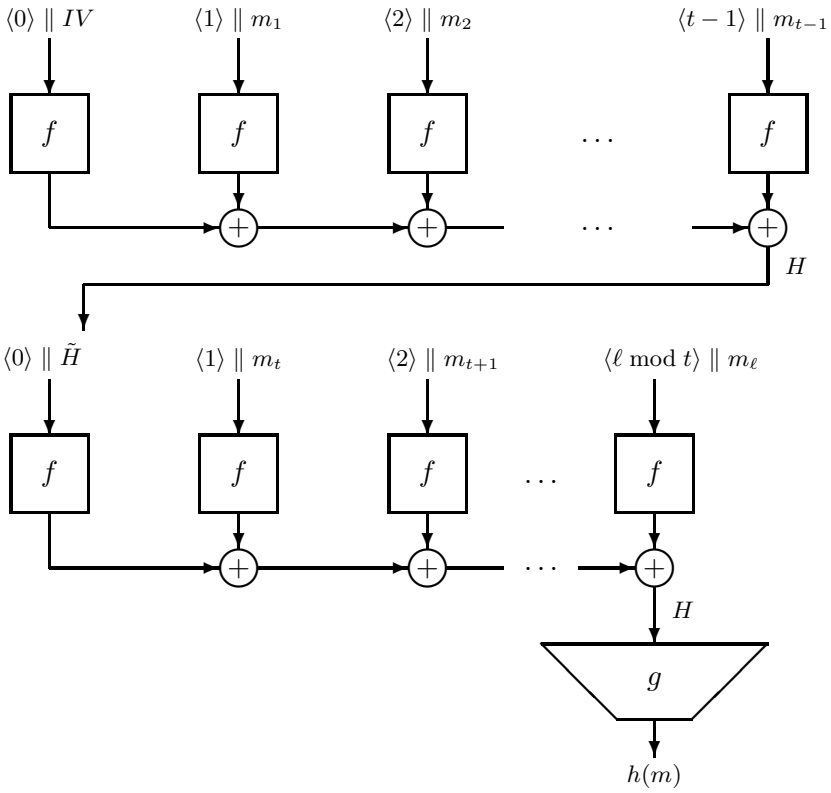



FIG. 9-2 – XOR-Hash - Description graphique.

égal à 2. Afin de simplifier la comparaison avec des constructions itératives classiques, nous choisissons de définir le taux de hachage de notre construction comme pour une fonction de compression avec un nombre fixe $t$ de combinaisons.

Puisque $r$ est de l'ordre de quelques centaines de bits et que $t$ varie de 2 à 8, on a $(r - \lceil \log_2 t \rceil) \approx r$. En conséquence :

$$\mathrm{Hr}_f \approx \frac{t}{t-1},$$

ce qui constitue un taux de hachage efficace.

**Construction hybride.** Dans la section relative aux considérations de design, nous avons souligné le fait que dans le but de résister à l'attaque du paradoxe des anniversaires généralisé, la taille de la variable de chaînage se doit d'être de l'ordre de quelques centaines de bits. Une telle construction itérative munie d'une fonction de compression fondée sur le paradigme de randomisation puis combinaison peut dès lors, être inefficace pour des messages courts. Par exemple, les paramètres $r = 640$ et $t = 8$ conduisent à une fonction de compression nécessitant un bloc de message de taille supérieure ou égale à 4480 bits. La construction hybride permet d'obtenir une fonction efficiente y compris pour des message courts. Nous montrons de plus, dans la section suivante, que cette construction n'affaiblit pas le schéma. En effet, les

hypothèses sur lesquelles réside la sécurité de notre construction prennent en compte le principe de la construction hybride.

### 9-1.3 Analyse de sécurité

Nous considérons dans cette section la sécurité de notre construction. Nous discutons tout d'abord des notions de fonctions de hachage idéales et de fonction de hachage sûres. Nous introduisons ensuite, les 6 problèmes sur lesquels reposent la sécurité de XOR-Hash.

**Fonction de hachage sûre et fonction de hachage idéale.** Une fonction de hachage idéale $h$ produisant des empreintes de taille $n$ bits se doit de vérifier les propriétés classiques exigées des fonctions de hachage cryptographiques. Les attaques par recherche d'antécédent et de second antécédent doivent nécessiter $\mathcal{O}(2^n)$ évaluations de cette fonction, $\mathcal{O}(2^{n/2})$ évaluations pour une attaque par collision. Nous remarquons que dans le livre référence : *Handbook of Applied Cryptography* [MOV96], la notion de fonction de hachage cryptographique sûre requiert seulement que ces attaques soient hors de portée d'un adversaire (*Computationally Infeasible*). Bien évidemment une fonction de hachage idéale constitue une fonction de hachage sûre. Cependant une fonction de hachage qui ne vérifie pas de façon idéale les propriétés de résistance au calcul d'antécédent et de second antécédent, peut tout de même constituer une fonction de hachage sûre.

Ainsi l'objectif de sécurité que nous nous fixons est de produire une fonction de hachage sûre. Le coût des attaques classiques doit être au delà de la puissance de calcul d'un adversaire.

**Problèmes.** Dans le but d'estimer la sécurité de notre construction, nous définissons une liste des problèmes que doit résoudre un attaquant afin de mener à bien une attaque contre la fonction XOR-Hash.

**Problème 1 ($t$-sum preimage pour $f$)** Étant donné $s \in \{0,1\}^r$, trouver $x_1, \ldots, x_l$ distincts appartenant à $\{0,1\}^r$ avec $0 < l \leq t$ tels que $f(x_1) \oplus \ldots \oplus f(x_l) = s$.

**Problème 2 ($t$-sum collision pour $f$)** Trouver $x_1, \ldots, x_l$ distincts appartenant à $\{0,1\}^r$ avec $0 < l \leq 2t$ tels que $f(x_1) \oplus \ldots \oplus f(x_l) = 0$.

**Problème 3 ($t$-sum 2nd preimage pour $f$)** Étant donnés $x_1, \ldots, x_l$ appartenant à $\{0,1\}^r$ avec $l \leq t$, trouver $x'_1, \ldots, x'_{l'}$ distincts appartenant à $\{0,1\}^r$ avec $0 < l' \leq t$ tels que $f(x_1) \oplus \ldots \oplus f(x_l) = f(x'_1) \oplus \ldots \oplus f(x'_{l'})$.

**Problème 4 (preimage pour $g$)** Étant donné $y \in \{0,1\}^n$, trouver $x \in \{0,1\}^r$ tel que $g(x) = y$.

**Problème 5** Trouver $s \in \{0,1\}^r$ et $x_1, \ldots, x_l$ distincts appartenant à $\{0,1\}^r$ avec $0 < l \leq t$ tels que $f(x_1) \oplus \ldots \oplus f(x_l) \neq s$ et $g(f(x_1) \oplus \ldots \oplus f(x_l)) = g(s)$.

**Problème 6** Étant donné $s \in \{0,1\}^r$, trouver $x_1, \ldots, x_l$ distincts appartenant à $\{0,1\}^r$ avec $0 < l \leq t$ tels que $f(x_1) \oplus \ldots \oplus f(x_l) \neq s$ et $g(f(x_1) \oplus \ldots \oplus f(x_l)) = g(s)$.

Les Problèmes 1, 2 et 3 se réduisent à différentes instances des problèmes de décodage de syndrome et de décodage de syndrome double sous l'hypothèse que la fonction $f$ est une fonction pseudo-aléatoire. En effet, résoudre ces problèmes

revient à décoder un syndrome pour une matrice aléatoire de dimensions $(r, 2^r)$. Ces problèmes ont été traités dans l'article de Augot *et al.* [AFS05].

Le Problème 4 pose comme exigence pour la fonction $g$ de résister au calcul d'antécédent.

Les Problèmes 5 et 6 impliquent les deux fonctions $f$ et $g$. Dans une certaine mesure, ils expriment une certaine forme d'"indépendance" entre $f$ et de $g$ :

- La résistance au Problème 5 induit une résistance aux collisions pour $g$ relativement à $f$. Il ne suffit pas à un attaquant de pouvoir trouver une collision "aléatoire" (indépendamment de $f$) pour $g$, pour être capable de casser le schéma.
- La résistance au Problème 6 induit une résistance au calcul de second antécédent pour $g$ relativement à $f$. Il ne suffit pas à un attaquant de pouvoir trouver un second antécédent "aléatoire" (indépendamment de $f$) pour $g$, pour être capable de casser le schéma.

**Arguments de sécurité.** On remarque facilement que si les fonctions $f$ et $g$ sont des fonctions pseudo-aléatoires, la construction que nous proposons est sûre. La définition des problèmes que nous présentons à pour but de spécifier les propriétés minimales que nous exigeons de ces fonctions pour garantir la sécurité du schéma. Nous ne sommes pas parvenus à construire formellement des réductions de sécurité. Cependant, nous conjecturons que la sécurité de la fonction XOR-Hash se réduit aux hypothèses suivantes :

1. Nous conjecturons que les résistances aux attaques par collisions, par calcul d'antécédent et par calcul de second antécédent de notre fonction de hachage se réduisent à l'un des six problèmes énumérés précédemment.
2. Nous présumons que la meilleure attaque contre les Problèmes 1, 2 et 3 est l'algorithme du paradoxe des anniversaires généralisé de Wagner. Cette attaque possède une complexité au moins égale à $O\left(2^{r/(1+\log_2 t)}\right)$ pour les Problèmes 1 et 3, et une complexité au moins égale à $O\left(2^{r/(2+\log_2 t)}\right)$ pour le Problème 2.
3. Nous présumons que $g$ est telle que les Problèmes 4 et 6 sont au moins aussi difficiles que les Problèmes 1 à 3.

Sous ces hypothèses, notre construction fournit au moins $r/(2+\log_2 t)$ bits de sécurité contre les attaques par collision. Nous choisissons donc en conséquence le paramètre $n$ (la taille de la sortie de $g$) égal à $2r/(2+\log_2 t)$. Nous présentons table 9-1 quelques choix de paramètres de sécurité sûrs pour la résistance aux collisions.

### 9-1.4 Conclusion

Nous avons présenté une nouvelle construction de fonction de hachage cryptographique. La sécurité de cette construction est liée à la difficulté, pour une fonction à sens-unique donnée $f : \{0,1\}^r \to \{0,1\}^r$, à trouver un ensemble de valeurs $x_1, \ldots, x_l$ avec $l \leq 2t$ telles que la somme $f(x_1) \oplus \cdots \oplus f(x_l)$ soit égale à une certaine valeur ; pour un paramètre $t$ fixé. Ce type de problème est apparenté à des problèmes de théorie des codes réputés durs.

Nous envisageons de continuer à rechercher une réduction formelle de sécurité pour la fonction XOR-Hash.

| sécurité | $r$ | $t$ | $n$ | taux de hachage |
|---|---|---|---|---|
| $2^{80}$ | 240 | 2 | 160 | $\approx 2$ |
| $2^{80}$ | 320 | 4 | 160 | $\approx 4/3$ |
| $2^{128}$ | 384 | 2 | 256 | $\approx 2$ |
| $2^{128}$ | 512 | 4 | 256 | $\approx 4/3$ |
| $2^{128}$ | 640 | 8 | 256 | $\approx 8/7$ |
| $2^k$ | $k \times (2 + \ell)$ | $2^\ell$ | $2k$ | $\approx t/(t-1)$ |

TAB. 9-1 – Paramètres de sécurité résistants aux attaques par collision pour la fonction XOR-Hash.

## 9-2 La fonction FSB

La première publication de la fonction de hachage FSB a été présentée en 2005 [AFS05]. Les auteurs de cette publication ont continué à collaborer sur cette fonction afin d'en améliorer les performances et les preuves de sécurité. Ce travail continu à conduit à la version élaborée en réponse à la compétition organisée par le NIST en 2008 [AFG08], dont l'intégralité de la documentation est disponible à l'adresse : *http ://www-rocq.inria.fr/secret/CBCrypto/index.php ?pg=fsb*.

Bien que notre contribution à ce processus soit des plus modeste, nous reprenons ici les caractéristiques principales de la fonction FSB.

### 9-2.1 Description

La fonction FSB (pour *Fast Syndrome Based hash function*) est une fonction de hachage itérative utilisant l'algorithme de Merkle-Damgård en conjonction avec une fonction de compression principale possédant un état interne large et une fonction de compression secondaire appliquée lors d'une itération finale.

La fonction de compression principale prend comme argument une chaîne de bits de taille $s$ et produit en sortie une chaîne de bits de taille $r$. Elle utilise des techniques de théorie des codes pour calculer le syndrome d'un mot binaire, dérivé du bloc de message et de la variable de chaînage en entrée, pour produire la nouvelle variable de chaînage. En pratique, la sortie de cette fonction est constituée du $XOR$ de permutations circulaires d'un ensemble de vecteurs prédéfinis. L'ensemble de tous les vecteurs possibles constitue les colonnes d'une matrice $H$ de taille $r \times n$. Le processus interne à cette fonction consiste donc à générer un mot binaire particulier (appelé mot régulier) de longueur $n$ et de poids de Hamming $w$ puis à le multiplier par la matrice $H$. Du point de vue de la théorie des codes, ce mot de poids $w$ peut être interprété comme un motif d'erreur, et la multiplication par la matrice $H$ comme le calcul du syndrome de cette erreur. La fonction de compression secondaire est construite à partir de la fonction Whirlpool [BR00]. Lors de la dernière itération de la fonction de compression principale, une variable de chaînage de taille $r$ bits est obtenue. Cette variable est alors à son tour compressée par l'application de la fonction de hachage Whirlpool pour produire un haché de taille 512 bits. Ce haché

| Dénomination de la version | Taille de l'empreinte | Paramètres | |
|---|---|---|---|
| | | $s$ | $r$ |
| $FSB_{160}$ | 160 bits | 1120 | 640 |
| $FSB_{224}$ | 224 bits | 1568 | 896 |
| $FSB_{256}$ | 256 bits | 1792 | 1024 |
| $FSB_{384}$ | 384 bits | 2392 | 1472 |
| $FSB_{512}$ | 512 bits | 3224 | 1984 |

TAB. 9-2 – Paramètres pour les 5 versions de la fonction de hachage FSB.

est alors tronqué pour produire une empreinte pour le message correspondant à la taille désirée.

La fonction de hachage cryptographique FSB soumise au NIST se décline en 5 versions que nous résumons dans le tableau 9-2.

### 9-2.2 Sécurité

Les meilleures attaques contre la fonction de compression principale sont bien identifiées et il est possible d'évaluer précisément leur complexité. Le tableau 9-3 donne les complexités des différentes attaques qui ont été prises en compte lors de la sélection des paramètres pour la fonction de hachage FSB. Le choix de l'utilisation de l'algorithme de Merkle-Damgård en tant qu'extenseur de domaine et de la fonction Whirpool en tant que fonction de compression finale ont été fait dans le but de préserver autant que possible la sécurité de la fonction de compression principale. Si nous notons $CF$ cette fonction de compression et $HF$ la fonction de hachage dans son ensemble, nous avons :

1. pour la résistance aux collisions, l'algorithme de Merkle-Damgård garantit que $b$ bits de sécurité pour $CF$ induisent $b$ bits de sécurité pour $HF$,
2. pour la résistance au calcul d'antécédent, $b$ bits de sécurité pour $CF$ induisent aussi $b$ bits de sécurité pour $HF$,
3. pour la résistance au calcul de second antécédent, du fait de l'attaque générique de Kelsey et Schneier [KS05], $b$ bits de sécurité pour $CF$ induisent seulement $b - k$ bits de sécurité pour $HF$.

Finalement, pour un paramètre $n \in \{160, 224, 256, 384, 512\}$ et un message de longueur au plus $2^k$, les meilleures attaques connues contre la fonction$FSB_n$ sont des attaques génériques avec les complexités suivantes :

- recherche de collisions : $2^{n/2}$,
- recherche d'antécédent : $2^n$,
- recherche de second antécédent : $2^{n-k}$,

### 9-2.3 Conclusion

La fonction de hachage cryptographique FSB s'est révélée plus lente que la plupart de ses concurrentes à la compétition SHA-3. De plus, la taille nécessaire à sa

| | Collision | | Preimage | | 2nd Preimage | |
|---|---|---|---|---|---|---|
| | $GBA$ $(n, 2w, r)$ | $ISD$ $(n, 2w, r)$ | $GBA$ $(n, w, r)$ | $ISD$ $(n, w, r)$ | $GBA$ $(n-w, w, r)$ | $ISD$ $(n-w, w, r)$ |
| $FSB_{160}$ | $2^{119.2}$ | $2^{100.3}$ | $2^{163.6}$ | $2^{211.1}$ | $2^{163.6}$ | $2^{211.1}$ |
| $FSB_{224}$ | $2^{166.9}$ | $2^{135.3}$ | $2^{229.0}$ | $2^{292.0}$ | $2^{229.0}$ | $2^{292.0}$ |
| $FSB_{256}$ | $2^{190.7}$ | $2^{153.0}$ | $2^{261.6}$ | $2^{330.8}$ | $2^{261.6}$ | $2^{330.8}$ |
| $FSB_{384}$ | $2^{281.0}$ | $2^{215.5}$ | $2^{391.5}$ | $2^{476.7}$ | $2^{391.5}$ | $2^{476.7}$ |
| $FSB_{512}$ | $2^{378.7}$ | $2^{285.6}$ | $2^{527.4}$ | $2^{687.8}$ | $2^{527.4}$ | $2^{687.8}$ |

TAB. 9-3 – Complexité des meilleures attaques connues contre la fonction de compression principale de FSB. La notation $GBA$ indique l'algorithme du paradoxe des anniversaires généralisé, la notation $ISD$ indique l'algorithme de décodage par ensemble d'information.

description est de l'ordre de 2Mb, ce qui peut constituer un désavantage pour les implémentations en hardware. Elle constitue cependant, la fonction de hachage la plus rapide pour laquelle il existe des réductions de sécurité strictes vers un problème algorithmique difficile.

## Quatrième partie

# Conclusions et perspectives

# 10

# Conclusions et perspectives

## Conclusions

La majorité des années de cette thèse a été consacrée à l'étude des fonctions de hachage SHA-0 et SHA-1, et plus particulièrement aux cryptanalyses de ces fonctions. L'analyse de ces cryptanalyses a nécessité un fort investissement en terme d'énergie et de temps ; cet investissement à toutefois porté ses fruits. D'un point de vue pratique, nous avons avec Thomas Peyrin publié en 2008 une attaque contre la fonction SHA-0, qui reste à ce jour la meilleure attaque par collision publiée pour cette fonction. Concernant la fonction SHA-1, un travail plus théorique, centré sur la caractéristique linéaire, a aboutit à une classification des vecteurs de perturbations ainsi qu'à la mise à jour de comportements non répertoriés de certains enchevêtrements de collisions locales.

La seconde partie du travail mené l'a été sous la forme de la conception avec Nicolas Sendrier de la fonction XOR-Hash. La fonction proposée constitue un modèle de conception de type coquille vide (*Empty Shell Construction*), dont nous avons relié la sécurité à un certain nombre de problèmes que nous avons définis. Nous avons été de plus invité à collaborer au processus qui à conduit à la proposition de la fonction de hachage FSB en réponse à la compétition organisée par le NIST.

Durant la dernière année de cette thèse, dans le cadre de l'emploi d'un poste d'ATER, nous avons avec l'équipe IA de l'université Paris 8 produit un article relatif à la colorisation d'images segmentées accepté pour la conférence SITIS 2010. Cette incursion dans le domaine de la synthèse d'image (bien éloigné de celui des fonctions de hachage cryptographiques) nous a permit de constater que les compétences de recherche acquises durant cette thèse pouvaient être mises au service d'autres domaines de l'informatique.

## Perspectives

Comme nous avons pu le détailler dans cette thèse, une attaque par collision contre la fonctions SHA-1 nécessite la mise en oeuvre de modèles, de techniques et d'outils variés. Le modèle de collision locale introduit par Chabaud et Joux et la construction de vecteurs de perturbations constitue sans doute la meilleure approche dont nous disposons. La génération de caractéristiques non-linéaires, bien que représentant une difficulté certaine relativement à son implémentation en tant qu'outil

automatisé, est aujourd'hui une technique relativement bien maîtrisée. La technique des blocs multiples est bien comprises, et les techniques d'accélération de recherche, modifications de message et bits neutres, ont été formalisées et réunies dans le modèle des attaques boomerang.

Cependant, nous ne possédons toujours pas de collision pour l'intégralité des 80 pas de la fonction SHA-1. En effet, la complexité d'une telle attaque dépend finalement de façon très directe du vecteur de perturbations utilisé. Les meilleures attaques pratiques dont nous disposons sont des attaques permettant d'obtenir des collisions pour des versions réduites à 70 et 73 pas, où les vecteurs de perturbations employés sont optimisés spécifiquement pour ce nombre de pas. Par exemple, le vecteur utilisé dans [DCM07] qui conduit à une collision sur 70 pas pour une complexité de $2^{44}$ conduirait à une complexité de $2^{88}$ (supérieure au coût de l'attaque générique) pour couvrir les 10 pas restants. L'étude et la recherche de vecteurs de perturbations est donc d'un intérêt majeur pour l'obtention d'une collision pour la fonction SHA-1.

De plus, la majorité des analyses de comportement des collisions locales qui sont présentes dans la littérature reposent sur des hypothèses d'indépendance. Cependant, nombre de cas pathologiques ont été mis à jour au long des différentes cryptanalyses menées. Parmi ces cas, certains comme la compression de bit ont permis d'améliorer les performances, d'autres comme les chemins différentiels impossibles ont nécessités un traitement particulier. L'étude statistique du comportement des collisions locales met à jour l'existence de nouveaux cas pathologiques non répertoriés et conduit à l'observation de décalages entre les probabilités théoriques de bon comportement et les probabilités mesurées. Ces faits constituent une remise en cause de l'hypothèse d'indépendance dans un certain nombre de cas de figures. Une étude approfondie de ces phénomènes conduirait certainement à améliorer notre compréhension des mécanismes de propagation des collisions locales au sein des états des registres.

Les travaux que nous avons menés relativement à la génération de vecteurs de perturbations ont contribué à répondre à un certain nombre de questions et de remarques présentes dans la littérature, sous la forme d'une classification de ces vecteurs. Mais l'analyse des enchevêtrement de collisions locales soulève de nouvelles questions et remet en cause des positions établies :

- Les différences observées sur les probabilités de bon comportement des collisions locales ouvrent de nouvelles positions d'insertion : la position de bit 31 se révélant plus avantageuse dans certaines configurations.
- Le choix d'instanciation uniforme des signes des collisions locales pour un vecteur de perturbations demande à être justifié (même si en pratique il fonctionne correctement pour la position de bit 2 avec les vecteurs utilisés dans la littérature) puisque il apparaît que pour certains types d'enchevêtrement il conduit à des chemins différentiels impossibles.

Les pistes de recherche sur lesquelles nous souhaitons continuer à travailler consistent donc à poursuivre l'analyse du comportement des enchevêtrements de collisions locales afin de construire un modèle à même d'expliquer ces comportements et/ou d'en tirer partie pour la cryptanalyse.

Nous avons aussi, en collaboration avec Sunandan Chakraborty, travaillé sur un outil formel pour la génération de caractéristiques non-linéaires. Nous envisageons de continuer la mise au point de cet outil afin de mettre en oeuvre une recherche effective de collision pour les 80 pas de la fonction SHA-1.

# Bibliographie


[ABD09] E. Andreeva, C. Bouillaguet, O. Dunkelman, and J. Kelsey, Herding, second preimage and trojan message attacks beyond merkle-damgard. In M. J. Jacobson Jr., V. Rijmen and R. Safavi-Naini editors, *Selected Areas in Cryptography - SAC 2009*, *Lecture Notes in Computer Science 5867*, pages 393-414. Springer Verlag, 2009.

[ABF08] E. Andreeva, C. Bouillaguet, P.-A. Fouque, J. J. Hoch, J. Kelsey, A. Shamir and S. Zimmer. Second preimage attacks on dithered hash functions. In N. P. Smart editor, *Advances in Cryptology - EUROCRYPT 2008*, *Lecture Notes in Computer Science 4965*, pages 270-288. Springer Verlag, 2008.

[ANP07] E. Andreeva, G. Neven, B. Preneel and T. Shrimpton, Seven-Property-Preserving Iterated Hashing : ROX. In K. Kurosawa editor, *Advances in Cryptology - ASIACRYPT 2007*, *Lecture Notes in Computer Science 4833*, pages 130-146. Springer Verlag, 2007.

[AFS05] D. Augot, M. Finiasz and N. Sendrier, A Family of Fast Syndrome Based Cryptographic Hash Functions. In E. Dawson and S. Vaudenay editors, *Progress in Cryptology - MYCRYPT 2005*, *Lecture Notes in Computer Science 3715*, pages 64-83. Springer Verlag, 2005.

[AFG08] D. Augot, M. Finiasz, P. Gaborit, S. Manuel and N. Sendrier, Fast Syndrome-Based hash function. Available at *http ://www-rocq.inria.fr/secret/CBCrypto/index.php ?pg=fsb*. 2008.

[ADL08] Y. Arbitman, G. Dogon, V. Lyubashevsky, D. Micciancio, C. Peikert and A. Rosen, SWIFFTX : A Proposal for the SHA-3 Standard Available at *http ://www.eecs.harvard.edu/ alon/PAPERS/lattices/swifftx.pdf*. 2008.

[BR00] P. Barreto and V. Rijmen, The Whirlpool Hashing Function. Available at *http ://paginas.terra.com.br/informatica/paulobarreto/WhirlpoolPage.html*. First version in 2000 revised in May 2003.

[BGR95] M. Bellare, R. Guerin and P. Rogaway, XOR MACs : New Methods for Message Authentication Using Finite Pseudorandom Functions. In D. Coppersmith editor, *Advances in Cryptology - CRYPTO 1995*, *Lecture Notes in Computer Science 963*, pages 15-20. Springer Verlag, 1995.

[BM97] M. Bellare and D. Micciancio, A New Paradigm for Collision-free Hashing : Incrementality at Reduced Cost. In W. Fumy editor, *Advances in Cryptology - EUROCRYPT 1997*, *Lecture Notes in Computer Science 963*, pages 15-20. Springer Verlag, 1997.

[BR06] M. Bellare and T. Ristenpart, Multi-Property-Preserving Hash Domain Extension and the EMD Transform. In X. Lai and K. Chen editors, *Advances in*

*Cryptology - ASIACRYPT 2006*, *Lecture Notes in Computer Science 4284*, pages 229-314. Springer Verlag, 2006.

[BR93] M. Bellare and P. Rogaway, Random Oracles are Practical : A paradigm for Designing Efficient Protocols. *ACM Conference on Computer and Communications Security 1993*, pages 62-73. ACM, 1993.

[Ber07a] D. J. Bernstein, The Salsa20 Family of Stream Ciphers. Available at *http ://cr.yp.to/papers.html#salsafamily*. 2007.

[Ber07b] D. J. Bernstein, What Output Size Resists Collisions in a XOR of Independent Expansions ? In proceedings of *ECRYPT Hash Workshop 2007*, pages 66-75. 2007.

[Ber92] T. A. Berson, Differential Cryptanalysis Mod $2^{32}$ with Applications to MD5. In R. A. Rueppel editor, *Advances in Cryptology - EUROCRYPT 1992*, *Lecture Notes in Computer Science 658*, pages 71-80. Springer Verlag, 1993.

[BDP06] G. Bertoni, J. Daemen, M. Peeters and G. Van Assche, Cryptographic Sponges. Available at *http ://sponge.noekeon.org/*. 2006.

[BDP08a] G. Bertoni, J. Daemen, M. Peeters and G. Van Assche, On the Indifferentiability of the Sponge Construction. In N. P. Smart editor, *Advances in Cryptology - EUROCRYPT 2008*, *Lecture Notes in Computer Science 4965*, pages 181-197. Springer Verlag, 2008.

[BDP08b] G. Bertoni, J. Daemen, M. Peeters and G. Van Assche, KECCAK. Available at *http ://keccak.noekeon.org/*. 2008.

[BC04] E. Biham and R. Chen, Near-Collisions of SHA-0. In M. K. Franklin editor, *Advances in Cryptology - CRYPTO 2004*, *Lecture Notes in Computer Science 3152*, pages 290-305. Springer Verlag, 2004.

[BCJ05] E. Biham, R. Chen, A. Joux, P. Carribault, C. Lemuet and W. Jalby, Collisions of SHA-0 and Reduced SHA-1. In R. Cramer editor, *Advances in Cryptology - EUROCRYPT 2005*, *Lecture Notes in Computer Science 3494*, pages 36-57. Springer Verlag, 2005.

[BD06] E. Biham and O. Dunkelman, A Framework for Iterative Hash Functions : HAIFA. In proceedings of *Second NIST Cryptographic Hash Workshop*. Available at *http ://csrc.nist.gov/groups/ST/hash/second_ workshop.html*. 2006.

[BS90] E. Biham, A. Shamir, Differential Cryptanalysis of DES-like Cryptosystems. In A. Menezes and S. A. Vansto editors, *Advances in Cryptology - CRYPTO 1990*, *Lecture Notes in Computer Science 537*, pages 2-21. Springer Verlag, 1991.

[BS91a] E. Biham, A. Shamir, Differential Cryptanalysis of Feal and N-Hash. In D. W. Davies editor, *Advances in Cryptology - EUROCRYPT 1991*, *Lecture Notes in Computer Science 547*, pages 1-16. Springer Verlag, 1991.

[BS91b] E. Biham, A. Shamir, Differential Cryptanalysis of Snefru, Khafre, REDOC-II, LOKI and Lucifer. In J. Feigenbaum editor, *Advances in Cryptology - CRYPTO 1991*, *Lecture Notes in Computer Science 576*, pages 156-171. Springer Verlag, 1992.

[BRS02] J. Black, P. Rogaway and T. Shrimpton, Black-Box Analysis of the Block-Cipher-Based Hash Function Constructions from PGV. In M. Yung editor, *Advances in Cryptology - CRYPTO 2002*, *Lecture Notes in Computer Science 2442*, pages 320-335. Springer Verlag, 2002.

[BCC08] E. Bresson, A. Canteaut, B. Chevallier-Mames, C. Clavier, T. Fuhr, A. Gouget, T. Icart, J. Misarsky, M. Naya-Plasencia, P. Paillier, T. Pornin, J. Reinhard, C. Thuillet and M. Videau, SHABAL. Available at *http ://www.shabal.com/*. 2008.

[BAC08] D. R. L. Brown, A. Antipa, M. Campagna and R. Struik, ECOH : the Elliptic Curve Only Hash Available at *http ://ehash.iaik.tugraz.at/uploads/a/a5/Ecoh.pdf*. 2008.

[CGH98] R. Canetti, O. Goldreich and S. Halevi, The random oracle methodology, revisited. *STOC'1998*, pages 209-218. ACM, 1998.

[CC98] A. Canteaut and F. Chabaud, New Algorithm for Finding Minimum-Weight Words in a Linear Code : Application to McEliece's Cryptosystem and to Narrow-Sense BCH Codes of Length 511. *IEEE Transactions on Information Theory*, 44(1), pages 376-378. 1998.

[CJ98] F. Chabaud and A. Joux, Differential Collisions in SHA-0. In H. Krawczyk editor, *Advances in Cryptology - CRYPTO 1998*, *Lecture Notes in Computer Science 1462*, pages 56-71. Springer Verlag, 1998.

[CJ98] S. Contini, A. K. Lenstra and R. Steinfeld, VSH, an Efficient and Provable Collision-Resistant Hash Function. In S. Vaudenay editor, *Advances in Cryptology - EUROCRYPT 2006*, *Lecture Notes in Computer Science 4004*, pages 165-182. Springer Verlag, 2006.

[CPM90] D. Coppersmith, S. Pilpel, C. H. Meyer, S. M. Matyas, M. M. Hyden, J. Oseas, B. Brachtl and M Schilling, Data Authentication Using Modification Detection Codes Based on a Public One Way Encryption Function. U.S. Patent No. 4,908,861, Filed August 28, 1987. 1990

[CDM05] J. S. Coron, Y. Dodis, C. Malinaud and P. Puniya, Merkle-Damgard Revisited : How to Construct a Hash Function. In V. Shoup editor, *Advances in Cryptology - CRYPTO 2005*, *Lecture Notes in Computer Science 3621*, pages 430-448. Springer-Verlag, 2005.

[Dam89] I. Damgard, A Design Principle for Hash Functions. In G. Brassard editor, *Advances in Cryptology - CRYPTO 1989*, *Lecture Notes in Computer Science 435*, pages 416-427. Springer Verlag, 1990.

[Dea99] R. D. Dean, Formal Aspects of Mobile Code Security. PhD thesis, Princeton University. Available at http ://www.cs.princeton.edu/sip/pub/ddean-dissertation.php3. 1999.

[DBB91] B. den Boer and A. Bosselaers, An Attack on the Last Two Rounds of MD4. In J. Feigenbaum editor, *Advances in Cryptology - CRYPTO 1991*, *Lecture Notes in Computer Science 765*, pages 293-304. Springer Verlag, 1992.

[DBB93] B. den Boer and A. Bosselaers, Collisions for the Compression Function of MD5. In T. Helleseth editor, *Advances in Cryptology - EUROCRYPT 1993*, *Lecture Notes in Computer Science 765*, pages 293-304. Springer Verlag, 1994.

[DCM07] C. De Cannière, F. Mendel and C. Rechberger, Collisions for 70-step SHA-1 : On the Full Cost of Collision Search. In C. Adams, A. Miri and M. J. Wiener editors, *Selected Areas in Cryptography - SAC 2007*, *Lecture Notes in Computer Science 4876*, pages 56-73. Springer-Verlag, 2007.

[DCR06] C. De Canière and C. Rechberger, Finding SHA-1 Characteristics : General Results and Applications. In X. Lai and K. Chen editors, *Advances in Cryptology - ASIACRYPT 2006*, *Lecture Notes in Computer Science 4284*, pages 1-20. Springer Verlag, 2006.

[DCR08] C. De Canière and C. Rechberger, Preimages for Reduced SHA-0 and SHA-1. In D. Wagner editor, *Advances in Cryptology - CRYPT0 2008*, *Lecture Notes in Computer Science 5157*, pages 179-202. Springer Verlag, 2008.

[DH76] W. Diffie and M. E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, *IT-22(6)*, pages 644-654. 1976.

[Dob96a] H. Dobbertin, Cryptanalysis of MD4. In D. Gollmann editor, *Fast Software Encryption - FSE 1996*, *Lecture Notes in Computer Science 1039*, pages 53-69. Springer-Verlag, 1996.

[Dob96b] H. Dobbertin, Cryptanalysis of MD5 compress. In Rump Session of *Advances in Cryptology - EUROCRYPT 1996*. Available at *ftp ://ftp.rsasecurity.com/pub/cryptobytes/crypto2n2.pdf*. 1996.

[FIPS-AES] National Institute of Standards and Technology, FIPS 187 : Advanced Encryption Standard. Available at *http ://csrc.nist.gov*. 2001.

[FIPS-DES] National Institute of Standards and Technology, FIPS 46-3 : Data Encryption Standard. Available at *http ://csrc.nist.gov*. 1999.

[FIPS-SHA0] National Institute of Standards and Technology, FIPS 180 : Secure Hash Standard. Available at *http ://csrc.nist.gov*. 1993.

[FIPS-SHA1] National Institute of Standards and Technology, FIPS 180-1 : Secure Hash Standard. Available at *http ://csrc.nist.gov*. 1995.

[FIPS-SHA2a] National Institute of Standards and Technology, FIPS 180-2 : Secure Hash Standard. Available at *http ://csrc.nist.gov*. 2002.

[FIPS-SHA2b] National Institute of Standards and Technology, FIPS 180-2 : Secure Hash Standard - Change notice 1. Available at *http ://csrc.nist.gov*. 2004.

[FIPS-SHA3] National Institute of Standards and Technology, FIPS 180-3 : Secure Hash Standard. Available at *http ://csrc.nist.gov*.

[Gre10] E. A. Grechnikov, Collisions for 72-step and 73-step SHA-1 : Improvements in the Method of Characteristics *Cryptology ePrint Archive*, Report 2010/413. Available at *http ://eprint.iacr.org/2010/413*. 2010.

[HN04] H. Handsuch and D. Naccache, SHACAL. NESSIE book, Final Report of European project IST-1999-12324 Available at *https ://cosic.esat.kuleuven/nessie/Bookv015.pdf*. 2004.

[Jou04] A. Joux, Multi-collisions in Iterated Hash Functions. Application to Cascaded Constructions. In M. Franklin editor, *Advances in Cryptology - CRYPTO 2004*, *Lecture Notes in Computer Science 3152*, pages 306-316. Springer-Verlag, 2004.

[JP07] A. Joux and T. Peyrin, Hash Functions and the (Amplified) Boomerang Attack. In A. Menezes editor, *Advances in Cryptology - CRYPTO 2007*, *Lecture Notes in Computer Science 4622*, pages 244-263. Springer-Verlag, 2007.

[JP05] C.S. Jutla and A. C. Patthak, A Matching Lower Bound on the Minimum Weight of SHA-1 Expansion Code. *Cryptology ePrint Archive*, Report 2005/266. Available at *http ://eprint.iacr.org/2005/266*. 2005.

[KKS00] J. Kelsey, T. Khono and B. Schneier, Amplified Boomerang Attacks Against Reduced-Round MARS and Serpent. In B. Schneier editor, *Fast Software Encryption - FSE 2000*, *Lecture Notes in Computer Science 1978*, pages 75-93. Springer Verlag, 2001.

[KS05] J. Kelsey and B. Schneier, Second Preimages on n-Bit Hash Functions for Much Less than $2^n$ Work. In R. Cramer editor, *Advances in Cryptology - EUROCRYPT 2005*, *Lecture Notes in Computer Science 3494*, pages 474-490. Springer Verlag, 2005.

[KK06] J. Kelsey and T. Khono, Herding Hash Functions and the Nostradamus Attack. In S. Vaudenay editor, *Advances in Cryptology - EUROCRYPT 2006*, *Lecture Notes in Computer Science 4004*, pages 183-200. Springer Verlag, 2006.

[KL94] L. R. Knudsen and X. Lai, New Attacks on All Double Block Length Hash Functions of Hash Rate 1, Including the Parallel-DM. In A. De Santis editor, *Advances in Cryptology - EUROCRYPT 1994*, *Lecture Notes in Computer Science 950*, pages 410-418. Springer Verlag, 1994.

[KP96] L. R. Knudsen and B. Preneel, Hash Functions Based on Block Ciphers and Quaternary Codes. In K. Kim and T. Matsumoto editors, *Advances in Cryptology - ASIACRYPT 1996*, *Lecture Notes in Computer Science 1163*, pages 77-90. Springer Verlag, 1996.

[KP97] L. R. Knudsen and B. Preneel, Fast and Secure Hashing Based on Codes. In B. S. Kalinski Jr. editor, *Advances in Cryptology - CRYPTO 1997*, *Lecture Notes in Computer Science 1294*, pages 485-498. Springer Verlag, 1997.

[KP02] L. R. Knudsen and B. Preneel, Construction of Secure and Fast Hash Functions Using Nonbinary Error-Correcting Codes. *IEEE Transactions on Information Theory*, 48(9), pages 2524-2539. 2002.

[LM92] X. Lai and J. L. Massey, Hash Functions Based on Block Ciphers. In R. A. Rueppel editor, *Advances in Cryptology - EUROCRYPT 1992*, *Lecture Notes in Computer Science 658*, pages 55-70. Springer Verlag, 1992.

[Leo88] J.S. Leon, A Probabilistic Algorithm for Computing the Minimum Weight of Large Error-Correcting Codes. *IEEE Transactions on Information Theory*, 34(5), pages 1354-1359. 1988.

[Luc05] S. Lucks, A Failure-Friendly Design Principle for Hash Functions. In B. K. Roy editor, *Advances in Cryptology - ASIACRYPT 2005*, *Lecture Notes in Computer Science 3788*, pages 474-494. Springer Verlag, 2005.

[Man06] S. Manuel, Cryptanalyses différentielles de SHA-0. Mémoire pour l'obtention du Mastère Recherche, Mathématiques Applications au Codage et à la Cryptologie. Université Paris8. Available at *http ://www-rocq.inria.fr/secret/Stephane.Manuel*. 2008.

[Man08] S. Manuel, Classification and Generation of Disturbance Vectors for Collision Attacks against SHA-1. *Cryptology ePrint Archive*, Report 2008/469. Available at *http ://eprint.iacr.org/2008/469*. 2008.

[Man09] S. Manuel, Classification and Generation of Disturbance Vectors for Collision Attacks against SHA-1. In A. Kolosha, E. Rosnes and M. Parker editors, *International Workshop on Coding and Cryptography - WCC 2009*, pages 224-235. 2009.

[Man10] S. Manuel, Classification and Generation of Disturbance Vectors for Collision Attacks against SHA-1. To appear in*Design, Codes and Cryptography - DCC 2010*. 2010.

[MP08] S. Manuel and T. Peyrin, Collisions on SHA-0 in one hour. In K. Nyberg, editor, *Fast Software Encryption - FSE 2008*, *Lecture Notes in Computer Science 5086*, pages 16-35. Springer-Verlag, 2008.

[MS07] S. Manuel and N. Sendrier, XOR-Hash : A Hash Function Based on XOR. *Western European Workshop on Research in Cryptology - WEWoRC 2007*. 2007.

[MP05] K. Matusiewicz and J. Pieprzyk, Finding Good Differential Patterns for Attacks on SHA-1. In O. Ytrehus editor, *International Workshop on Coding and Cryptography - WCC 2005*, *Lecture Notes in Computer Science 3969*, pages 164-177. Springer-Verlag, 2005.

[MRH04] U. Maurer, R. Renner, and C. Holenstein, Indifferentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology. In M. Naor editor, *Theory of Cryptography - TCC 2004*, *Lecture Notes in Computer Science 2951*, pages 21-39. Springer Verlag, 2004.

[MPR06] F. Mendel, N. Pramstaller, C. Rechberger and V. Rijmen, The Impact Of Carries on the Complexity of Collision Attacks on SHA-1. In M. J. B. Robshaw editor, *Fast Software Encryption - FSE 2006*, *Lecture Notes in Computer Science 4047*, pages 278-292. Springer Verlag, 2006.

[MRR07] F. Mendel, C. Rechberger and V. Rijmen, Update on SHA-1. In Rump Session of *Advances in Cryptology - CRYPTO 2007*. Available at http ://rump2007.cr.yp.to/09-rechberger.pdf. 2007.

[MOV96] A. J. Menezes, P. C. Van Oorschot and S.A. Vanstone, Handbook of Applied Cryptography. CRC Press, 1996.

[Mer79a] R. C. Merkle, Secrecy, Authentication, and Public Key Systems. PhD thesis, Stanford University. 1979.

[Mer79b] R.C. Merkle, Method of providing digital signatures. U.S. Patent No. 4,309,569. 1980.

[Mer89] R.C. Merkle, One Way Hash Function and DES. In G. Brassard editor, *Advances in Cryptology - CRYPTO 1989*, *Lecture Notes in Computer Science 435*, pages 428-446. Springer Verlag, 1989.

[Mer90] R.C. Merkle, A Fast Software One-Way Hash Function. *Journal of Cryptology*, 3(1), pages 43-58. Springer Verlag, 1990.

[MS88] C. H. Meyer and M. Schilling, Secure program load with manipulation detection code. Securicom 88, pages 111-130. SEDEP, 1988.

[MOI90] S. Miyaguchi, K. Ohta, and M. Iwata, 128-bit hash function (N-hash). *NTT Review*, 2(6), pages 128-132. Springer Verlag, 1990.

[NSS06] Y. Naito, Y. Sasaki, T. Shimoyama, J. Yajima, N. Kunihiro and K. Otha, Improved Collision Search for SHA-0. In X. Lai and K. Chen editors, *Advances in Cryptology - ASIACRYPT 2006*, *Lecture Notes in Computer Science 4284*, pages 21-36. Springer-Verlag, 2006.

[Nil02] J. B. Nielsen, Separating Random Oracle Proofs from Complexity Theoretic Proofs : The Noncommitting Encryption Case. In M. Yung editor, *Advances in*

*Cryptology - CRYPT0 2002*, *Lecture Notes in Computer Science 2442*, pages 111-126. Springer-Verlag, 2002.

[Pey08] T. Peyrin, Analyse de fonctions de hachage cryptographiques. PhD thesis, Université de Versailles Saint-Quentin-en-Yvelines. Available at http ://sites.google.com/site/thomaspeyrin/research. 2008.

[PGM06] T. Peyrin, H. Gilbert, F. Muller and M. J. B. Robshaw, Combining Compression Functions and Block Cipher-Based Hash Functions. In X. Lai and K. Chen editors, *Advances in Cryptology - ASIACRYPT 2006*, *Lecture Notes in Computer Science 4284*, pages 315-331. Springer Verlag, 2006.

[PRR05] N. Pramstaller, C. Rechberger and V. Rijmen. Exploiting Coding Theory for Collision Attacks on SHA-1. In N.P. Smart editor, *Cryptography and Coding 2005*, *Lecture Notes in Computer Science 3796*, pages 78-95. Springer-Verlag, 2005.

[Pre93] B. Preneel, Analysis and Design of Cryptographic Hash Functions, PhD thesis, Katholieke Universiteit Leuven. Available at https ://www.cosic.esat.kuleuven.be/publications/ 1993.

[PB95] B. Preneel and A. Bosselaers, RIPE, Integrity Primitives for Secure Information Systems. *Final Report of RACE Integrity Primitives Evaluation (RIPE-RACE 1040)*, *Lecture Notes in Computer Science 107*. Springer Verlag, 1995.

[PGV93] B. Preneel, R. Govaerts and J. Vandewalle, Hash Functions Based on Block Ciphers : A Synthetic Approach. In D. R. Stinson editor, *Advances in Cryptology - CRYPTO 1993*, *Lecture Notes in Computer Science 773*, pages 368-378. Springer Verlag, 1993.

[PVO96] B. Preneel and P. C. van Oorschot, On the Security of Two MAC Algorithms. In U. M. Maurer editor, *Advances in Cryptology - EUROCRYPTO 1996*, *Lecture Notes in Computer Science 1070*, pages 19-32. Springer Verlag, 1996.

[Rab78] M. O. Rabin, Digitalized signatures. In R. Lipton and R. DeMillo editors, *Foundations of Secure Computations*, pages 155-168. Academic Press, 1978.

[RSY10] M. R. Reyhanitabar, W. Susilo and Yi Mu, Enhanced Security Notions for Dedicated-Key Hash Functions : Definitions and Relationships. In S. Hong and T. Iwata editors, *Fast Software Encryption - FSE 2010*, *Lecture Notes in Computer Science 6147*, pages 192-211. Springer Verlag, 2010.

[RO05] V. Rijmen and E. Oswald, Update on SHA-1. In A. J. Menezes editor, *The Cryptographers'Track at the RSA conference - CT-RSA 2005*, *Lecture Notes in Computer Science 3376*, pages 58-71. Springer-Verlag, 2005.

[RFC-MD2] J. Linn, RFC 1115 : Privacy Enhancement for Internet Electronic Mail : Part III – Algorithms, Modes, and Identifiers Available at *http ://www.ietf.org/rfc/rfc1115.txt*. 1989.

[RFC-MD4] R. L. Rivest, RFC 1186 : The MD4 Message-Digest Algorithm. Available at *http ://www.ietf.org/rfc/rfc1320.txt*. 1990.

[RFC-MD5] R. L. Rivest, RFC 1321 : The MD5 Message-Digest Algorithm. Available at *http ://www.ietf.org/rfc/rfc1321.txt*. 1992.

[RAB08] R. Rivest, B. Agre, D. V. Bailey, C. Crutchfield, Y. Dodis, K. E. Fleming, A. Khan, J. Krishnamurthy, Y. Lin, L. Reyzin, E. Shen, J. Sukha, D. Sutherland, E.

Tromer and Y. L. Yin, MD6. Available at *http ://groups.csail.mit.edu/cis/md6/*. 2008.

[Rog95] P. Rogaway, Bucket Hashing and its Application to Fast Message Authentication. In D. Coppersmith editor, *Advances in Cryptology - CRYPTO 1995*, *Lecture Notes in Computer Science 963*, pages 29-42. Springer Verlag, 1995.

[Rog06] P. Rogaway, Formalizing Human Ignorance. In P. Q. Nguyen editor, *Progress in Cryptology - VIETCRYPT 06*, *Lecture Notes in Computer Science 4341*, pages 211-228. Springer Verlag, 2006.

[RS04] P. Rogaway and T. Shrimpton, Cryptographic Hash-Function Basics : Definitions, Implications and Separations for Preimage Resistance, Second-Preimage Resistance and Collision Resistance. In B. K. Roy and W. Meier editors, *Fast Software Encryption - FSE 2004*, *Lecture Notes in Computer Science 3017*, pages 371-388. Springer Verlag, 2004.

[SKP07] M. Sugita, M. Kawazoe, L. Perret and H. Imai, Algebraic Cryptanalysis of 58-Round SHA-1. In A. Biryukov editor, *Fast Software Encryption - FSE 2007*, *Lecture Notes in Computer Science 4593*, pages 349-365. Springer-Verlag, 2007.

[Wag99] D. Wagner, The Boomerang Attack. In L. R . Knudsen editor, *Fast Software Encryption - FSE 1999*, *Lecture Notes in Computer Science 1636*, pages 156-170. Springer-Verlag, 1999.

[Wag02] D. Wagner, A Generalized Birthday Problem. In M. Yung editor, *Advances in Cryptology - CRYPTO 2002*, *Lecture Notes in Computer Science 2442*, pages 288-303. Springer Verlag, 2002.

[Wan97] X. Wang, The Collision Attack on SHA-0. Previously available at *http ://www.infosec.edu.cn*. 1997.

[Wan98] X. Wang, The Improved Collision Attack on SHA-0. Previously available at *http ://www.infosec.edu.cn*. 1998.

[WFL04] X. Wang, D. Feng, X. Lai and H. Yu, Collisions for Hash Functions MD4, MD5, HAVA-128 and RIPEMD. *Cryptology ePrint Archive*, Report 2004/199. Available at *http ://eprint.iacr.org/2004/199.pdf*. 2004.

[WLF05] X. Wang, X. Lai, D. Feng, H. Chen and X. Yu, Crytanalysis of the Hash Functions MD4 and RIPEMD. In R. Cramer editor, *Advances in Cryptology - EUROCRYPT 2005*, *Lecture Notes in Computer Science 3494*, pages 1-18. Springer Verlag, 2005.

[WY05] X. Wang and H. Yu, How to break MD5 and other hash functions. In R. Cramer editor, *Advances in Cryptology - EUROCRYPT 2005*, *Lecture Notes in Computer Science 3494*, pages 19-35. Springer Verlag, 2005.

[WYY05a] X. Wang, A. C. Yao and F. Yao, Cryptanalysis on SHA-1. In proceedings of *NIST Cryptographic Hash Workshop*. Available at *http ://csrc.nist.gov*. 2005.

[WYY05b] X. Wang, X. L. Yin and H. Yu, Finding Collisions in the Full SHA-1. In V. Shoup editor, *Advances in Cryptology - CRYPTO 2005*, *Lecture Notes in Computer Science 3621*, pages 17-36. Springer Verlag, 2005.

[WYY05c] X. Wang, X. L. Yin and H. Yu, New Collision Search for SHA-1. In Rump Session of *Advances in Cryptology - CRYPTO 2005*. 2005.

[WYY05d] X. Wang, H. Yu and X. L. Yin, Efficient Collision Search Attacks on SHA-0. In V. Shoup editor, *Advances in Cryptology - CRYPTO 2005*, *Lecture Notes in Computer Science 3621*, pages 1-16. Springer Verlag, 2005.

[YSN07] J. Yajima, Y. Sasaki, Y. Naito, T. Iwasaki, T. Shimoyama, N. Kunihiro, and K. Ohta, A new strategy for finding a differential path of sha-1. *Information Security and Privacy - ISP 2007*, *Lecture Notes in Computer Science 4586* pages 45-58. Springer Verlag, 2007.

[YIN08] J. Yajima and T. Iwasaki and Y. Naito and Y. Sasaki and T. Shimoyama and N. Kunihiro and K. Ohta, A Strict Evaluation Method on the Number of Conditions for the SHA-1 Collision Search. In M. Abe and V. Gligor editors, *ACM Symposium on Information, Computer and Communication Security - ASIACCS 2008*, pages 10-20. ACM, 2008.

[YIN09] J. Yajima and T. Iwasaki and Y. Naito and Y. Sasaki and T. Shimoyama and T. Peyrin and N. Kunihiro and K. Ohta, A Strict Evaluation Method on the Number of Conditions for the SHA-1 Collision Search. *IEICE Trans. Fundamentals E92-A(1)*, pages 87-95. IEICE, 2009.

# Table des figures

# Liste des tableaux

# Bibliographie

## Article de journal

[DCC 2010] S. Manuel, Classification and Generation of Disturbance Vectors for Collision Attacks against SHA-1. Á paraître dans *Design, Codes and Cryptography - DCC 2010*. 2010.

## Colloques avec comité de sélection et actes

[SITIS 2010] C. Sauvaget, S. Manuel, JN. Vittaut, J. Suarez et V. Boyer, Segmented Images Colorization Using Harmony. À paraître dans les actes de *Signal Image Technology and Internet Based Systems - SITIS 2010*. 2010

[WCC 2009] S. Manuel, Classification and Generation of Disturbance Vectors for Collision Attacks against SHA-1. Parus dans les actes de *International Workshop on Coding and Cryptography - WCC 2009*. 2009.

[FSE 2008] S. Manuel et T. Peyrin, Collisions on SHA-0 in one hour. Éditeur K. Nyberg, *Fast Software Encryption - FSE 2008*, *Lecture Notes in Computer Science 5086*, pages 16-35. Springer-Verlag, 2008.

## Colloques avec comité de sélection sans actes

[WEWoRC 2007] S. Manuel et N. Sendrier, XOR-Hash : A Hash Function Based on XOR. *Western European Workshop on Research in Cryptology - WEWoRC 2007*.

## Colloques sans comité de sélection

[C2 2009] S. Manuel, Attaques par collision contre SHA-1. *Journées Codage et Cryptographie - C2 2009*

[C2 2008] S. Manuel, Produire une collision pour SHA-0 en une heure. *Journées Codage et Cryptographie - C2 2008*

## Autre publication

[FSB] D. Augot, M. Finiasz, P. Gaborit, S. Manuel and N. Sendrier, Fast Syndrome-Based hash function. Disponible à l'adresse *http ://www-rocq.inria.fr/secret/CBCrypto/index.php ?pg=fsb*. 2008.

# Résumé

Une fonction de hachage est une fonction prenant comme argument un élément de taille arbitraire finie et renvoyant un élément de longueur fixée. Il existe différents types de fonctions de hachage qui correspondent à autant de domaines d'utilisation. Parmi ces fonctions, les fonctions de hachage cryptographiques se distinguent par la variété des missions qui leur sont confiées et par l'exigence qui leur est faîte de respecter de nombreux impératifs de sécurité. Les fonctions de hachage cryptographiques les plus utilisées en pratiques appartiennent à la famille MD-SHA, dont les membres les plus connus sont les fonctions MD5 et SHA-1. Durant ces dernières années, de nouvelles techniques de crytptanalyses ont fait leur apparition. Ces techniques, bien que très complexes, se sont montrés si efficaces qu'elles ont conduit à l'abandon de l'utilisation des fonctions MD5 et SHA-1, et à l'ouverture d'une compétition internationale pour le développement d'un nouvel algorithme de hachage cryptographique.

Les travaux de recherche que nous avons menés dans le cadre de cette thèse s'inscrivent à la fois dans une démarche d'analyse et de conception. Nous étudions les nouvelles avancées dans la cryptanalyse des fonctions de hachage, et plus particulièrement leurs mise en oeuvre dans le cadre des fonctions SHA-0 et SHA-1. Nous présentons à ce titre la meilleure attaque pratique connue à ce jour contre SHA-0 et proposons la première classification des vecteurs de perturbations utilisés par les attaques par collision contre la fonction SHA-1. Nous abordons ensuite la conception de nouvelle fonctions par le biais des fonction XOR-Hash et FSB.



**Mots-clés:** cryptographie, fonctions de hachage, cryptanalyse, SHA, XOR-Hash, FSB



# Abstract

A hash function is a function taking as argument an element of finite arbitrary length and returning an element of fixed length. There are different types of hash functions that correspond to fields of use. Among these functions, cryptographic hash functions are distinguished by the variety of missions assigned to them and requiring them to meet many security requirements. Cryptographic hash functions commonly used in practices belong to the MD-SHA family, whose most-known members are MD5 and SHA-1. In recent years, new crytptanalysis techniques have emerged. These techniques, although very complex, have proved so successful that it led to the abandonment of the use of MD5 and SHA-1, and to the openning of an international competition to develop a new cryptographic hash algorithm.

The research we conducted as part of this thesis stand in both process analysis and design. We study the new advances in the cryptanalysis of hash functions, particularly their implementation within the functions SHA-0 and SHA-1. We present as such the best practical attack known against SHA-0 and propose the first classification of disturbances vectors used by collision attacks against SHA-1. We then discuss the design of new functions through XOR-Hash and FSB.



**Keywords:** Cryptography, Hash Functions, Cryptanalysis, SHA, XOR-Hash, FSB