Quelques résultats en cryptographie symétrique, pour les modèles de confiance dans les réseaux ambiants et la sécurité dans les réseaux de capteurs sans fil

Marine Minier

Université Claude Bernard – Lyon 1 **INSA de Lyon**
Numéro d'Ordre : HDR 2012 005

Mémoire d'Habilitation à Diriger des Recherches

présenté et soutenu publiquement le 31 Mai 2012 devant

l'Université Claude Bernard Lyon 1
et
l'INSA de Lyon

par

Marine MINIER

Quelques résultats en cryptographie symétrique, pour les modèles de confiance dans les réseaux ambiants et la sécurité dans les réseaux de capteurs sans fil

Composition du jury

Rapporteurs :	Willi Meier, FHNW (Suisse) Refik Molva, Institut Eurecom Nice François-Xavier Standaert, Université de Louvain-La-Neuve (Belgique)
Examinateurs :	Thierry Berger, Université de Limoges Anne Canteaut, INRIA Paris-Rocquencourt Henri Gilbert, ANSSI Isabelle Guérin-Lassous, Université Lyon 1 Tanguy Risset, INSA de Lyon

Centre d'Innovation en Télécommunications et Intégration de Services (Laboratoire CITI)

Remerciements

Je suis extrêmement reconnaissante à Willi Meier d'avoir accepté d'être rapporteur de ce mémoire. Ce manuscrit doit beaucoup à ses nombreux résultats concernant les chiffrements à flot. Il n'y aurait peut-être pas eu de compétition eStream sans les attaques algébriques et donc beaucoup des résultats de ce mémoire n'existeraient pas sans ses travaux.

Je tiens à remercier Refik Molva d'avoir accepté d'être rapporteur de ce manuscrit, d'autant plus que ce n'est pas la première fois que je le mets à contribution car il a déjà été rapporteur de la thèse de Wassim Znaidi. Sa notoriété internationale dans les domaines de la sécurité des réseaux, de la confiance et de la privacy en fait un exemple bien dur à suivre pour qui souhaite relever les défis de ces domaines de recherche.

Je remercie François-Xavier Standaert d'avoir accepté d'être rapporteur de ce manuscrit. J'ai beaucoup apprécié mes visites au Crypto Group de Louvain-La-Neuve et je le remercie pour la confiance qu'il m'a témoignée en me demandant d'être rapporteuse de la thèse de Baudoin Collard aux côtés d'éminents cryptographes tels qu'Eli Biham ou Vincent Rijmen.

Merci à Thierry Berger d'avoir accepté d'être membre de ce jury. Je préfère ne pas compter le nombre d'années depuis lesquelles je le connais en tant qu'élève tout d'abord puis en tant que doctorante. C'est grâce à lui et à François Arnault que j'ai découvert le monde fascinant des registres à décalage. Depuis, nos nombreux échanges, l'encadrement de la thèse de Benjamin Pousse, nos travaux communs et mes séjours à Limoges m'ont permis de comprendre à quel point c'est un chercheur inventif. Je lui dois beaucoup.

Merci également à Anne Canteaut d'avoir accepté d'être membre de ce jury d'autant que c'est finalement elle qui a fait le plus long voyage. Je la remercie également pour son chaleureux accueil au projet alors CODES durant les années 2003-2005 et pour son soutien constant tant à travers les projets RAPIDE ou BLOC que dans notre travail commun, le seul chiffrement à flot au nom imprononçable.

Merci également à Henri Gilbert d'avoir accepté d'être membre de ce jury. Il a également eu la lourde tâche d'encadrer ma thèse et je tiens à le remercier de m'avoir appris ce qu'était le travail, l'intuition et l'intelligence de la recherche.

Merci à Tanguy Risset d'avoir accepté de représenter l'INSA de Lyon au sein de ce jury. Avec Tanguy, nous avons été recrutés la même année au laboratoire CITI de l'INSA de Lyon, lui en tant que professeur et moi en tant que maître de conférences. Nous avons donc découvert ensemble les nombreux acronymes de l'INSA et ses traboules administratives. Nos nombreux échanges sur l'implémentation matérielle des FCSRs m'ont été d'une aide précieuse. Merci enfin à Isabelle Guérin-Lassous d'avoir accepté de représenter l'Université Lyon 1 au sein de ce jury.

La recherche n'existerait pas sans ses laboratoires et leurs membres. Je tiens donc à remercier tous mes collègues des différents projets et des différents laboratoires dans lesquels j'ai eu l'occasion de travailler. Merci donc à Lars Knudsen du DTU de Lyngby. Merci à Claude Carlet, Philippe Guillot et Sihem Mesnager de l'Université Paris 8. Merci à Daniel Augot, Jean-Pierre Tillich, Nicolas Sendrier, Marion Videau, Andrea Röck, Yann Laigle-Chapuy, Cédric Lauradoux, Céline Blondeau, Cristina Boura du projet d'abord CODES puis SECRET. Et enfin merci à tous les membres du laboratoire CITI anciens ou actuels et tout particulièrement à Fabrice Valois pour nos travaux communs et une fois encore à Cédric Lauradoux.

Je tiens également à remercier Stéphane Ubéda grâce à qui mon intégration au laboratoire CITI s'est faite dans les meilleures conditions possibles en me confiant, à mon arrivée, la responsabilité du projet KAA et le co-encadrement de la thèse de Samuel Galice. Je tiens également à le remercier car il m'a permis de comprendre pourquoi l'imitation peau d'autruche est plus jolie en violet.

Même si nous n'avons jamais co-signé d'article, je tiens à remercier tout particulièrement Pascale Charpin qui suit mes recherches depuis le début de ma thèse. Nous avons participé ensemble à de nombreux projets et nos discussions, notamment sur ce que doit être la recherche publique, ont été et sont toujours passionnantes.

Je dois également beaucoup aux doctorants que j'ai eu la chance d'encadrer. Ils rendent la recherche toujours plus passionnante. Par ordre d'apparition donc, je remercie Samuel Galice, Wassim Znaidi, Benjamin Pousse, Ochirkhand Erdene-Ochir et Gaël Thomas.

Je tiens également à remercier tous mes co-auteurs par ordre alphabétique : Anya Apavatjrut, François Arnault, Jean-Philippe Babau, Chérifa Boucetta, Nicolas Fournel, Antoine Fraboulet, Gérald Gavin, Mohamed Ali Kâafar, Apostolos A. Kountouris, Véronique Legrand, John Mullins, María Naya-Plasencia, Thomas Peyrin, Raphael Chung-Wei Phan, Yuanyuan Zhang ainsi que les coauteurs de SOSEMANUK et DECIM.

La vie à Lyon a été rendue moins grise grâce à de nouvelles rencontres et je les en remercie. Par ordre d'apparition donc : José, Sophie et Baptiste ; la bande des thésards du LIRIS/CITI ; Anaïs, Mathieu, Morgane et Alexis ; les voisins de l'immeuble d'en face ; Flavie pour sa chasse aux fautes d'orthographe dans le chapitre 3 ; les membres de l'atelier des canulars ; les aficionados des Caténors et de tiens bon la pente. Je tiens également à remercier les ami-e-s d'ailleurs toujours fidèles au poste : Alain, Anne et Benjamin, Blandine, Bruno, David et Cathy, Frédérique, Marc, Nadège, Sylvie et les ami-e-s de Vincent.

Je remercie le nouveau venu Walden pour le temps qu'il m'a permis de dégager vers quatre heures du matin.

Je remercie également et comme il se doit Vincent pour sa patience, sa chasse aux fautes d'orthographe et ses bons petits plats même si le poulet aux pruneaux a finalement fini sans pruneaux.

Au nouveau venu,
Walden.

Table des matières

Introduction générale

Le contenu de ce manuscrit pourra paraître quelque peu éclectique au lecteur peu averti. Il s'agit surtout d'un résumé de mes activités de recherche depuis 2004. J'ai d'une part souhaité conserver un pied dans les thèmes qui m'étaient chers auparavant, à savoir des activités concernant la cryptographie symétrique et d'autre part, mon intégration au sein du laboratoire CITI à partir de 2005 s'est faite vis à vis d'applications pour lesquelles se posaient des problématiques claires de sécurité. Il s'agissait donc des modèles de confiance pour les réseaux ambiants et de la sécurité dans les réseaux de capteurs, thème transverse du laboratoire qui lui vaut sa reconnaissance internationale.

Ce manuscrit suit donc naturellement ce découpage et traite, dans une première partie, des résultats relativement théoriques que j'ai obtenus depuis 2004 dans les différents domaines de la cryptographie symétrique tandis que la deuxième partie de ce mémoire s'attache à développer les solutions plus appliquées développées dans le domaine des modèles de confiance dans les réseaux ambiants et pour la sécurité dans les réseaux de capteurs.

Cryptographie symétrique

La cryptographie, signifiant littéralement écriture secrète, est donc la science qui s'attache à définir des mécanismes permettant de garantir un certain nombre de propriétés de sécurité comme la confidentialité, l'authenticité, l'intégrité, la signature mais également l'anonymat ou le vote électronique. Il existe deux formes de cryptographie : la cryptographie à clé secrète (ou symétrique) où pour que deux personnes puissent communiquer ensemble, elles doivent au préalable avoir échangé un secret commun et la cryptographie à clé publique (ou asymétrique) où cette condition n'est plus nécessaire car pour chiffrer un message à un destinataire, il suffit d'utiliser la clé publique de celui-ci et seul lui qui possède la clé secrète correspondant à cette clé publique pourra déchiffrer ce message. La cryptographie symétrique est utilisée relativement couramment depuis plusieurs centaines d'années tandis qu'il a fallu attendre les années 1970 pour voir apparaître la cryptographie à clé publique. Cette dernière forme de cryptographie restant cependant relativement lente, dans la pratique, elle sert à échanger de façon sûre une clé de chiffrement entre deux entités qui utiliseront ensuite la cryptographie symétrique durant leurs communications. On parle alors de cryptographie hybride.

Mes incursions dans la cryptographie asymétrique étant limitées à une participation à un article [GM09], la première partie de ce mémoire portera sur les recherches développées en cryptographie symétrique. La cryptographie symétrique s'intéresse à définir les primitives cryptographiques suivantes : les algorithmes de chiffrement qui permettent de garantir la confidentialité des échanges, les fonctions de hachage qui permettent de garantir l'intégrité d'un message et les codes d'authentification de messages (MAC) qui permettent de garantir l'intégrité et l'authenticité de données échangées. Bien sûr à partir de ces primitives, il est possible de définir des protocoles comme des protocoles d'identification ou d'authentification mais ce n'est pas mon propos ici.

Les algorithmes de chiffrement sont divisés en deux grandes catégories : les algorithmes de chiffrement à flot qui produisent une suite pseudo-aléatoire permettant de chiffrer à la volée un flot de données à l'aide d'une opération bijective élémentaire et les algorithmes de chiffrement par blocs qui, comme leur nom l'indique, découpent le message à chiffrer en blocs de taille fixe et chiffrent chaque bloc séparément en utilisant un mode de chiffrement. Les fonctions de hachage prennent en entrée un message de taille variable et produisent en sortie un condensé unique de taille fixe. Les codes d'authentification de messages (MAC) permettent de garantir à la fois l'intégrité et l'authenticité d'un message de taille variable en produisant à l'aide d'une clé un condensé de taille fixe.

Le chapitre 1 de la première partie recouvre l'ensemble des résultats que nous avons obtenus concernant les chiffrements à flot. Il s'agit essentiellement des résultats obtenus durant mon année d'ingénieur expert à l'INRIA Rocquencourt financé par le projet RNRT X-CRYPT [1] en 2004/2005 ainsi que de tous les résultats obtenus avec le laboratoire XLIM de l'Université de Limoges, dans le cadre de la thèse de Benjamin Pousse (2007-2010) et durant le projet ANR SETIN 2006 RAPIDE [2]. Ces derniers concernent les chiffrements à flot fondés sur les FCSR (Feedback with Carry Shift Register), les FCSR eux-même et les LFSR (Linear Feedback Shift Register). Le point de départ était ici l'utilisation de représentations simplifiées des FCSR et des LFSR via l'utilisation de matrices de transition. Cela a permis de mettre en lumière de nombreuses propriétés particulières utiles pour le design et l'implémentation optimisée de primitives cryptographiques.

Le chapitre 2 de la première partie décrit les résultats concernant les chiffrements par blocs, obtenus pour la plupart avec Raphaël Phan et Benjamin Pousse, et les résultats concernant les fonctions de hachage. En ce qui concerne les chiffrements par blocs, il s'agit essentiellement de cryptanalyses intégrales utilisées dans des modèles d'attaques particuliers relativement proches de ceux classiquement utilisés pour les fonctions de hachage. Ces modèles sont dits à clés connues ou à clés liées connues. L'actualité brûlante des fonctions de hachage liée à la compétition SHA-3 [3] a également orienté certaines de mes activités de recherche. La deuxième partie du chapitre 2 est donc consacrée à la construction de distingueurs contre des fonctions de hachage, il s'agit de distingueurs intégraux et d'attaques par rebond, ces dernières permettant de construire des collisions contre des candidats de la compétition SHA-3. Nous présentons également une construction de fonction de hachage ligthweight (c'est-à-dire avec une implémentation matérielle compacte) fondée sur un FCSR filtré linéairement et sur une construction éponge [BDPA08].

Confiance dans les réseaux ambiants et solutions de sécurité pour les réseaux de capteurs

La deuxième partie de ce manuscrit regroupe les travaux réalisés depuis mon arrivée au laboratoire CITI dans deux domaines : la confiance dans les réseaux ambiants et les solutions de sécurité que nous avons développées pour les réseaux de capteurs.

Le premier chapitre de cette partie est donc consacré au modèle de confiance fondé sur l'historique que nous avons développé dans le cadre du projet KAA [4] de l'appel ANR Sécurité informatique et de la thèse de Samuel Galice (2004-2007). Ce projet multi-disciplinaire avait pour but de définir un modèle de confiance socialement acceptable et informatiquement réalisable dans le contexte des réseaux ambiants où on considère que chaque utilisateur va pouvoir se

1. `http://www-roc.inria.fr/secret/X-Crypt/`
2. ANR SETIN (2007-2011) : `http://rapide-anr2006.gforge.inria.fr/`
3. `http://csrc.nist.gov/groups/ST/hash/sha-3/index.html`
4. ANR Sécurité Informatique (2004-2007) : `http://kaa.citi.insa-lyon.fr/`

connecter à son environnement ou à d'autres terminaux via son objet communiquant. Il s'agit donc de caractériser les liens que l'on souhaite construire dans des communautés ouvertes ou des communautés ayant des intérêts communs comme l'accès à un service particulier. Sur cette base, nous avons tenté d'intégrer au modèle que nous proposons des aspects sociologiques, juridiques et économiques.

Le deuxième chapitre de cette partie est consacré aux résultats que nous avons obtenus dans le domaine de la sécurité des réseaux de capteurs. Ces derniers peuvent être vus comme un nouveau paradigme entre les réseaux de surveillance et les réseaux ad-hoc. Il s'agit de réseaux multi-saut composés de petites entités appelées capteurs aux capacités et aux ressources limitées communiquant via le médium radio pour faire remonter des données, la plupart du temps environnementales, à un ou plusieurs points de collectes. De nombreuses attaques peuvent être considérées contre ce type de réseau, en raison des faibles capacités des éléments le constituant, du mode de routage multi-saut et du médium radio. La cryptographie permet de répondre correctement à un certain nombre d'attaques mais elle ne suffit pas à elle seule car les nœuds capteurs constituant le réseau peuvent être compromis. Dans ce cas, il est nécessaire de développer des solutions algorithmiques dédiées. C'est ce à quoi nous nous sommes employés dans le cadre de la thèse de Wassim Znaidi financée par la région Rhône-Alpes (2007-2010) où nous avons proposé plusieurs mécanismes de détection d'attaques particulières comme l'attaque *wormhole* ou l'attaque par réplication de nœuds. Dans le cadre du projet ANR ARESA2 [5] et de la thèse de Ochirkhand Erdene-Ochir (2009-2012), nous cherchons à construire des protocoles de routage intrinsèquement résilients à certains types d'attaques, c'est-à-dire que nous souhaitons définir des protocoles capables de continuer à router les messages en présence d'attaquants. Nous souhaitons, également, être capable de dégager des propriétés particulières de résilience permettant de renforcer les capacités du réseau au sens de cette notion. Enfin, dans le cadre du post-doctorat de Yuan Yuan Zhang et de la thèse de Wassim Znaidi, nous nous intéressons à la sécurité pour le codage réseau. Le codage réseau est une méthode récente permettant d'améliorer le débit de transmissions en mode multicast ou unicast via une dissémination d'information optimale. La plupart des solutions de sécurité développées dans ce cadre utilisent des méthodes de chiffrement ou de signature homomorphiques pour garantir la confidentialité de bout en bout et l'authenticité saut à saut ou des méthodes de routage multi-chemin.

5. ANR VERSO (2010-2013) : `http://aresa2.minalogic.net/`

Première partie

Cryptographie symétrique

Chapitre 1

Algorithmes de chiffrement à flot

1.1 Introduction

Les algorithmes de chiffrement à flot comme leur nom l'indique sont généralement utilisés pour chiffrer un flot de données à l'aide de la méthode du "one time pad". Il s'agit de produire une suite chiffrante de la taille du message qui sera xorée bit à bit (on peut également utiliser d'autres opérations bijectives sur des groupes plus gros que $\mathbb{F}_2$) avec chacun des bits du message, le processus de déchiffrement étant similaire à l'opération d'inversion près. Il faut donc pour cacher correctement le message que la suite chiffrante ressemble autant que faire se peut à une suite aléatoire afin de garantir que le chiffrement soit inconditionnellement sûr [Sha49].

C'est l'objet des chiffrements à flot : produire une suite pseudo-aléatoire ayant d'excellentes propriétés statistiques (longue période, propriété des runs [Gol81], etc.) à partir d'une clé partagée entre deux (ou plusieurs) entités. En d'autres termes, il s'agit de construire un générateur pseudo-aléatoire dont la valeur d'initialisation est secrète. Il est à noter cependant que si on initialise le générateur toujours à la même valeur, il produira systématiquement la même suite pseudo-aléatoire ne permettant pas de chiffrer deux messages distincts sans révéler de l'information à un attaquant. C'est pourquoi dans les applications pratiques, on initialise souvent un chiffrement à flot à l'aide de la clé K partagée et d'une valeur d'initialisation publique souvent notée IV différente pour chacun des messages à chiffrer. On parle dans ce cas de nonce au sens anglais de "number used once".

Il existe essentiellement deux grands types de chiffrements à flot :

- Les chiffrements à flot synchrones qui sont indépendants du message clair à chiffrer.
- Les chiffrements à flot auto-synchronisants qui produisent la suite chiffrante à partir du couple clé/vecteur d'initialisation et d'un nombre fixé d'éléments de la suite chiffrante déjà produits. On peut noter cependant que jusqu'à présent peu voire pas de chiffrements auto-synchronisants n'ont résisté à des attaques relativement basiques en raison peut-être de leur définition même. Le problème consistant à construire de tels chiffrements reste donc entier.

Dans le reste de ce chapitre, nous nous intéresserons essentiellement aux chiffrements à flot encore en vie à savoir les chiffrements à flot synchrones. Ces derniers sont utilisés surtout dans les trois contextes suivants : lorsque l'on souhaite chiffrer rapidement une grande quantité de données (chiffrement de disques durs par exemple), lorsque les ressources hardware sont très limitées (étiquettes RFID, etc.) et lorsque l'on utilise le médium radio car ces chiffrements ont le grand avantage de ne pas propager les erreurs (contrairement aux chiffrements par blocs). C'est cette dernière propriété qui a rendu leur utilisation si populaire dans les télécommunications.

On peut citer ici l'algorithme A5/1 utilisé dans la norme GSM, l'algorithme RC4 utilisé dans le protocole SSL ou l'algorithme E0 [Blu01] de Bluetooth.

On trouve en général dans ces chiffrements les éléments suivants (voir Fig. 1.1) :

- un registre à décalage fini faisant deux fois au moins la taille de la clé (afin de se prémunir contre les attaques par compromis temps/mémoire [HS05]) ;
- une fonction f de remise à jour de cet état ;
- une fonction g de filtrage non-linéaire choisie pour ses bonnes propriétés (équilibrée, haut degré, sans corrélation d'ordre 1, etc. [Can06]).

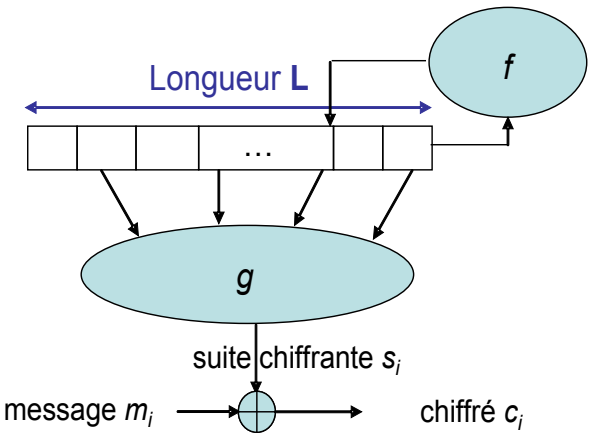


FIGURE 1.1 – Construction générique d'un chiffrement à flot.

Le chiffrement à flot lui-même utilise en général deux grandes étapes afin de produire la suite chiffrante à l'aide des trois composantes précédentes :

- le Key/IV setup qui produit, à partir de la clé secrète K et d'une valeur d'initialisation IV, un état initial pour le registre fini sans générer de sorties ;
- la génération de la suite chiffrante proprement dite (en général bit à bit ou groupe de bits par groupe de bits) qui remet à jour, à chaque top d'horloge, l'état du registre à l'aide de la fonction f et produit une sortie en filtrant l'état courant à l'aide de la fonction g.

Bien sûr, il existe d'autres moyens pour construire des chiffrements à flots comme la combinaison de registres (voir par exemple E0 [Blu01]), la décimation irrégulière de registres (voir le Shrinking Generator [CKM93]) ou la mise à jour de tables (voir RC4 [Riv92a]).

Jusqu'à l'apparition en 2003 des attaques algébriques [AF03, CM03, Cou03] dédiées aux chiffrements à flot, la fonction f de mise à jour du registre interne était le plus souvent linéaire et utilisait un registre à décalage à rétroaction linéaire, appelé en anglais Linear Feedback Shift Register (LFSR). Ce n'est plus le cas aujourd'hui. En effet, les attaques algébriques se fondent sur le fait que la relation liant l'état initial du registre interne (le plus souvent composé de la clé et de l'IV) avec l'état de ce registre après t clocks est linéaire. Ainsi, le système d'équations construit entre les bits de sortie du chiffrement et les bits de l'état du registre au temps t n'augmente pas de degré et il peut donc être résolu en utilisant des méthodes dédiées comme les bases de Gröbner [Fro98]. Des solutions existent également pour faire diminuer le degré du système par exemple en cherchant des multiples de g de bas degré (voir [CM03, Cou03, Can06]). Dans ce contexte, il devenait alors dangereux de continuer à utiliser des LFSR.

A partir de 2003, il a fallu repenser la manière dont on construisait habituellement les chiffre-

ments à flot et notamment la fonction f. La recherche dans ce domaine a été largement aidée par la compétition eStream [eST08] lancée en 2004 par le réseau d'excellence Européen ECRYPT [6] et terminée en 2008. Le but de cette compétition était de définir deux ensembles de chiffrements à flot qui pourraient être utilisés sans risques. La première catégorie concernait des algorithmes de chiffrement à flot dédiés aux applications logicielles sous des clés de taille d'au moins 128 bits pour un IV de 64 à 128 bits. La deuxième catégorie était dédiée à des chiffrements à flot hardware avec des clés de taille d'au moins 80 bits et des IV de 64 à 128 bits. Ainsi, six mois après son appel à primitives, eStream reçut 34 soumissions pour les deux catégories. Ce sont finalement 8 finalistes (4 dans la catégorie software et 4 dans la catégorie hardware) qui ont été retenus en mai 2008 après une phase d'intenses essais cryptanalytiques et d'évaluations de performances tant matérielles que logicielles. Notons que pour la plupart des finalistes, c'est l'utilisation d'au moins un registre à rétroaction non linéaire qui a prévalu comme dans Trivium [CP08] ou Grain [HJMM08] ou bien l'utilisation de composantes non-linéaires habituellement utilisées dans les algorithmes de chiffrement par blocs comme dans Sosemanuk ou Salsa20 [Ber08].

Cette compétition n'était pas sans risque car la précédente (NESSIE [NES01]) n'avait retenu aucun finaliste dans la catégorie chiffrement à flot. Seul SNOW devenant finalement SNOW v2 [EJ02] après une réparation a finalement survécu. Ce chiffrement à flot restait cependant une exception en 2004 car il utilise un LFSR, certes sur le corps $\mathbb{F}_{2^{32}}$ rendant par là même, les équations sur $\mathbb{F}_2$ non linéaires.

Durant mon année d'ingénieur expert au projet CODES de l'INRIA Rocquencourt financé par le projet RNRT XCRYPT, j'ai participé avec 13 co-auteurs à la soumission de deux primitives, une dans chaque catégorie. Il s'agissait de Sosemanuk [BBC+08a] pour la soumission logicielle et de Decim pour la soumission matérielle [BBC+08b]. Sosemanuk a été retenu comme finaliste tandis que Decim v2 a été éliminé après la phase 3 en raison essentiellement de ses performances matérielles insuffisantes. Sosemanuk combine des composants provenant à la fois de SNOW v2 [EJ02] et du chiffrement par blocs SERPENT [BAK98]. Il est relativement efficace à la fois en logiciel et en matériel [Can07, GB07]. Notons également qu'il reprend les principes démontrés par la suite dans [BG07] à savoir que les relations liant le premier bit de sortie à la clé et l'IV doivent être aussi complexes que dans le cas d'un chiffrement par blocs. Quant à Decim v2, il utilise un LFSR filtré dont la sortie est décimée par un mécanisme particulier appelé ABSG [GSB+05].

Parallèlement à ces deux soumissions, F. Arnault, T. Berger et C. Lauradoux ont soumis à la compétition eStream le chiffrement à flot F-FCSR-v2 [ABL06] avec une instance pour la catégorie logicielle et une instance pour la catégorie matérielle.

1.2 La saga des F-FCSR

La première version des F-FCSR a été proposé par F. Arnault et T. Berger à FSE 2005 [AB05b]. L'idée principale était de construire un chiffrement à flot relativement simple utilisant comme registre de base un FCSR (Feedback with Carry Shift Register, ou en français un registre à décalage à rétroaction avec retenue) filtré par une fonction linéaire sur $\mathbb{F}_2$. En effet, un FCSR est un registre linéaire sur $\mathbb{Z}_2$ (l'ensemble des entiers 2-adiques) et non plus sur $\mathbb{F}_2$, sa fonction de transition étant quadratique sur $\mathbb{F}_2$. Ces objets mathématiques ont été introduits dans les années 90 par A. Klapper et M. Goresky (voir par exemple [KG93, KG97] ; leur livre [GK09] est également un très bon résumé de leurs travaux concernant les LFSR, les FCSR et les AFSR (Algebraic Feedback Shift Registers).

6. voir `http://www.ecrypt.eu.org/`

Dans la suite de cette section avant de rentrer plus en avant dans la définition des F-FCSR, nous donnons quelques définitions concernant les FCSR rappelées de [KG93, Kob97, KG97, GK02, AB05a].

1.2.1 Quelques définitions

Un entier 2-adique s'écrit formellement à l'aide de son développement de Hensel $s = \sum_{i=0}^{\infty} s_i 2^i$, $s_i \in \{0, 1\}$. Cette série est convergente au sens de la topologie 2-adique. L'ensemble des entiers 2-adiques est noté $\mathbb{Z}_2$. L'addition et la multiplication dans $\mathbb{Z}_2$ sont faites en reportant les retenues sur les termes d'ordre supérieur, i.e. $2^n + 2^n = 2^{n+1}$ pour tout $n \in \mathbb{N}$. S'il existe un entier N tel que $s_n = 0$ pour tout $n \geq N$, alors s est un entier positif. Chaque entier impair q a un inverse dans $\mathbb{Z}_2$.

La propriété suivante donne une caractérisation complète des séquences binaires ultimement périodiques (voir [GK02] pour la preuve).

Proposition 1.2.1 *Soit $S = (s_n)_{n\in\mathbb{N}}$ une séquence binaire et soit $s = \sum_{i=0}^{\infty} s_i 2^i$ l'entier 2-adique correspondant. La séquence S est ultimement périodique si et seulement si, il existe deux nombres p et q dans $\mathbb{Z}$, q impair, tel que $s = p/q$.*
De plus, S est strictement périodique si et seulement si $pq \leq 0$ et $|p| \leq |q|$. Dans ce cas, on a la relation $s_n = (p \cdot 2^{-n} \mod q) \mod 2$.

La période de S est l'ordre de 2 modulo q, i.e., le plus petit entier T tel que $2^T \equiv 1 \pmod{q}$. La période satisfait $T \leq |q| - 1$. Si q est premier, alors T divise $|q| - 1$. Si $T = |q| - 1$, la séquence S est appelé une ℓ-séquence. Comme décrit dans [KG93, KG97, GK02, LR08], les ℓ-séquences ont beaucoup de propriétés prouvées très intéressantes qui peuvent être comparées à celles des m-séquences produites par des LFSR ayant un polynôme de rétroaction primitif : période connue, bonnes propriétés statistiques (équilibre, propriétés des runs, etc.), génération rapide, etc.

A partir de la proposition précédente, on peut définir un FCSR à l'aide d'un entier q premier et d'une valeur initiale p. L'automate produira alors la séquence infinie binaire s_i vérifiant la relation $p = q \cdot \sum_{i=0}^{\infty} s_i 2^i$. Plus précisément, si on définit la suite d'entiers $p(t)$ comme $p(0) = p$ et $p(t+1) = (p(t) - qs_i)/2$, les séquences (s_i) et $(p(t))$ peuvent être générées à partir d'un FCSR défini à l'aide de deux registres : un registre principal M et un registre de retenues C. Pour qu'un FCSR produise une ℓ-séquence, il faut que l'ordre de 2 soit maximal modulo q.

Ainsi, les FCSR produisant des ℓ-séquences partagent avec les LFSR produisant des m-séquences un certain nombre de leurs bonnes propriétés mais en plus garantissent une structure non linéaire sur $\mathbb{F}_2$.

En ce qui concerne la manière dont on représente les FCSR eux-même, il existe essentiellement deux grands types de représentations : la représentation en mode Galois et la représentation en mode Fibonacci. Dans [ABP11], F. Arnault, T. Berger et B. Pousse ont démontré comment les séquences produites par deux FCSR utilisant des représentations différentes mais définies par un même entier q sont équivalentes. Nous reviendrons par la suite sur ce résultat.

FCSR en mode Galois

Un FCSR en mode Galois (comme décrit dans la Fig. 1.2) est constitué d'un registre principal de n bits $M = (m_0, \ldots, m_{n-1})$ avec des positions de rétroaction fixées $d_0, \ldots, d_{n-1}$ où $d = \sum_{i=0}^{n-1} d_i 2^i = (1 + |q|)/2$. Toutes les rétroactions sont contrôlées par la cellule m_0 et par $n - 1$ cellules de retenues $C = (c_0, \ldots, c_{n-2})$. Au temps t, l'automate dans l'état (M, C) est mis à jour de la manière suivante :

1. Calcul de $x_i = m_{i+1} + c_i d_i + m_0 d_i$ pour tout i tel que $0 \leq i < n$ avec $m_n = 0$ et $c_{n-1} = 0$ et où m_0 représente le bit de rétroaction ;
2. Mise à jour de l'état courant : $m_i \leftarrow x_i \mod 2$ pour tout $i \in [0..n-1]$ et $c_i \leftarrow x_i \text{ div } 2$ pour $0 \leq i < n$ pour tout $i \in [0..n-2]$.

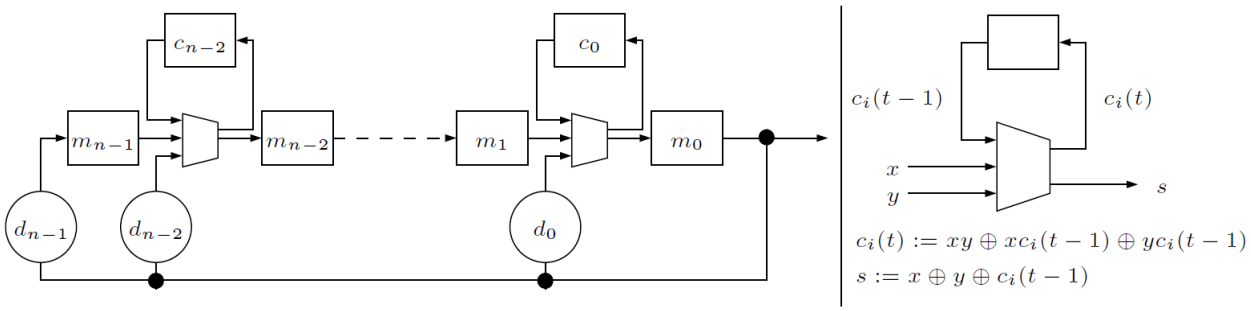


FIGURE 1.2 – Un FCSR en mode Galois et un additionneur 2 bits avec retenue.

FCSR en mode Fibonacci

Un FCSR en mode Fibonacci (comme décrit dans la Fig. 1.3) est composé d'un registre principal de n bits $M = (m_0, \ldots, m_{n-1})$. Les positions des rétroactions binaires $d = (d_0, \ldots, d_{n-1})$ sont associées à une retenue c calculée à partir de $w_H(d)$ cellules binaires où $w_H(d)$ est le poids de Hamming de $d = (1 + |q|)/2$.

Un automate dans l'état (M, c) est mis à jour de la manière suivante :

1. Calcul de $x = c + \sum_{i=0}^{n-1} m_i d_{n-1-i}$;
2. Mise à jour de l'état courant : $M \leftarrow (m_1, \ldots, m_{n-1}, x \mod 2)$, $c \leftarrow x \text{ div } 2$.

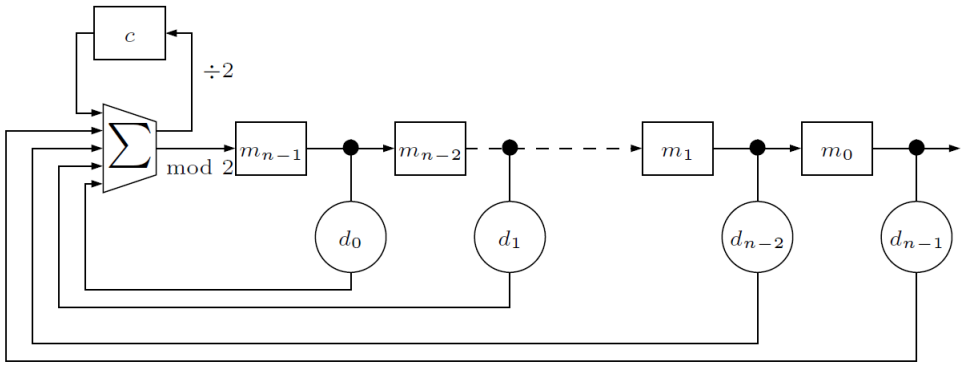


FIGURE 1.3 – Un FCSR en mode Fibonacci.

1.2.2 F-FCSR v1, F-FCSR v2 et X-FCSR v1

Dans [AB05b], les auteurs présentent plusieurs versions de F-FCSR v1 proposant donc d'utiliser un FCSR en mode Galois filtré par une fonction linéaire sur $\mathbb{F}_2$. L'article décrit également le processus de Key/IV setup.

Suite à cette publication, T. Berger et moi-même avons cherché quelles attaques nous pourrions monter contre cette première version des F-FCSR et les quatre instances décrites. Il est apparu assez rapidement que le nombre de clocks initialement proposé pour la phase de Key/IV setup était clairement insuffisant pour prévenir de façon efficace les attaques algébriques, le degré des équations liant les premiers bits de sortie avec les bits de clé et d'IV étant trop faible. Le nombre de monômes induits permettant de construire le système d'équations à résoudre dépendait également fortement de la connaissance ou non des valeurs de bits de retenues. Or, la première version de F-FCSR introduisait directement l'IV, une valeur qui peut être choisie par l'adversaire, dans le registre de retenues. Par exemple, il était possible d'attaquer F-FCSR-SF1 en utilisant 2^{15} valeurs d'IV connues avec une complexité de 2^{45} instructions binaires pour retrouver toute la clé. Ces résultats ont été publiés à Indocrypt 2005 [BM05].

Dans le même article, nous avons également donné des bornes théoriques et de simulations sur le nombre de clocks minimum qu'il fallait inclure durant la phase de Key/IV setup avant de sortir des bits pour faire en sorte que la complexité d'une attaque algébrique soit plus grande que celle d'une recherche exhaustive sur l'ensemble des clés possibles. A partir de 34 clocks, la complexité de l'attaque devient supérieure à la recherche exhaustive d'une clé de 128 bits. Ces résultats ont été confirmés par ceux obtenus avec B. Pousse et publiés au workshop SASC 2008 [PM08] permettant de dire que la construction du système d'équations algébriques devient extrêmement difficile après 10 itérations si à la fois les bits du registre principal et ceux du registre de retenues sont inconnus.

Parallèlement à ces résultats, E. Jaulmes et F. Muller à SAC 2005 [JM06] ont présenté deux autres attaques contre F-FCSR v1. La première utilise l'observation suivante : une différence bien choisie de 1 bit sur les valeurs d'IV n'a pas le temps en 6 clocks (le nombre d'itérations du Key/IV setup) de se propager correctement à tous les bits de sortie et permettra, en devinant 17 bits de la clé, de prédire la valeur de la différence sur le premier bit de sortie. Si cette différence est vérifiée, la supposition sur la valeur de la clé est bonne. En généralisant cette approche et en utilisant des fenêtres glissantes, les auteurs peuvent attaquer les quatre instances de F-FCSR v1 en utilisant 2^{15} valeurs d'IV pour une complexité maximale de l'ordre de 2^{32} opérations élémentaires. La deuxième attaque est une attaque par compromis temps/mémoire qui remarque que la taille du registre principal n'est pas suffisante car le nombre d'état réellement atteint après en moyenne 128 itérations est seulement de 2^{128}. Ainsi, une attaque par compromis temps/mémoire sera possible avec une complexité maximale de l'ordre de 2^{80} opérations en utilisant des données d'environ 2^{67} bits.

Afin de contrecarrer toutes les attaques précédentes, F. Arnault, T. Berger et C. Lauradoux ont finalement (après quelques aléas notamment liés à la mauvaise maîtrise des différences sur les retenues comme décrit dans [JM05]) proposé une deuxième version des F-FCSR soumise à l'appel eStream : F-FCSR v2 [ABL06] comportant deux instances dédiées. La première F-FCSR-H v2 est un chiffrement à flot construit pour le matériel et utilisant des clés de 80 bits. La deuxième F-FCSR-16 est dédiée au logiciel et supporte des clés de 128 bits. F-FCSR-H v2 a donc été retenu comme finaliste jusqu'en 2008 de la catégorie matérielle. Ce dernier se compose d'un FCSR de taille 160 bits défini par son premier de rétroaction q, d'un filtre linéaire construit à partir de la valeur de d permettant de sortir 8 bits à chaque clock et d'un processus de Key/IV setup. Ce dernier fonctionne de la manière suivante : le registre principal est initialisé à l'aide de la clé K et de la valeur IV tandis que le registre de retenues est initialisée à la valeur tout à 0 ; suit une phase de 20 itérations où le FCSR est clocké et à chaque clock, un octet est produit en sortie ; ces 20 itérations permettent de réinitialiser le FCSR ; puis ce dernier est clocké durant 162 itérations avant de commencer à sortir des octets.

Ce nombre d'itérations peut sembler ésotérique cependant, nous avons démontré avec F.

Arnault et T. Berger dans [ABM08] (cet article est également présenté en Annexe A) qu'elle est relativement bien choisie. En effet, si on s'intéresse au graphe de la fonction de transition d'un FCSR définie par un premier q de taille $n+1$ bits, celui-ci est composé de deux points fixes (0 et q) et d'un troisième composant contenant le reste des points, c'est-à-dire $2^{n+l}-2$ points où n est la taille du registre principal et où l vaut $W_H(d)-1$. De plus, cette dernière partie est constituée d'un cycle de taille $|q|-1$ et de radicelles convergeant vers ce cycle principal (voir l'exemple donné à la Fig. 1.4). Nous avons démontré dans [ABM08] que la taille maximale des radicelles était bornée par $n+4$ (même si la convergence en moyenne est beaucoup plus rapide). Dans ce même article, nous avons également démontré qu'un état où tous les bits de retenues sont à zéro ne peut pas être sur le cycle principal, c'est-à-dire qu'un état où toutes les retenues sont à zéro ne peut pas arriver dans F-FCSR v2 lorsque l'on a passé la phase de Key/IV setup. Ainsi, il est impossible de linéariser un FCSR en forçant à 0 toutes ses retenues.

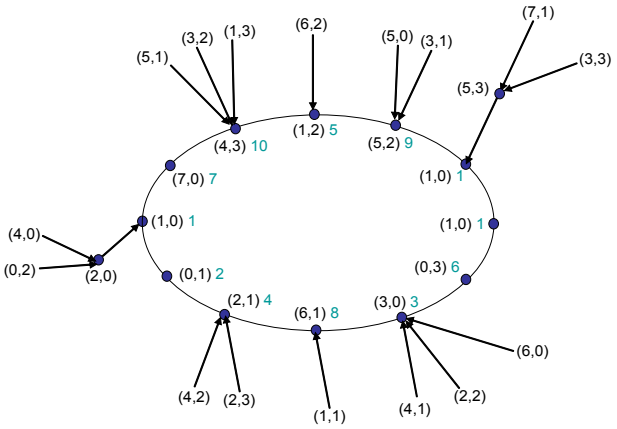


FIGURE 1.4 – Exemple de graphe pour un FCSR avec $q=-13$.

Notons également que fort de ce premier succès, nous avons présenté à Indocrypt 2007 [ABLM07] une version dédiée au logiciel d'un chiffrement à flot utilisant les FCSR : X-FCSR v1. Cette famille de chiffrement à flot est composée de deux instances : X-FCSR-128 qui permet de sortir à chaque clock 128 bits et X-FCSR-256, version plus risquée, qui permet de sortir à chaque clock 256 bits. La structure de cette famille de chiffrement est composée de :

- Deux FCSR en mode Galois de taille 256 bits décalés dans des sens oppposés ;
- Une fonction *Round* non-linéaire correspondant à un étage d'un chiffrement par blocs composée d'une série de boîtes S et d'une fonction de diffusion linéaire (celle-ci agit sur des mots de 128 bits pour X-FCSR-128 et sur des mots de 256 bits pour X-FCSR-256) ;
- 16 registres de mémoire de taille 128 bits pour X-FCSR-128 et 256 bits pour X-FCSR-256.

Ces éléments permettent de sortir 128 ou 256 bits à chaque clock de la manière suivante :

- **Phase de Key/IV setup** : comme proposé dans SOSEMANUK, cette partie n'utilise que la fonction *Round* appliquée à la valeur $IV \oplus K_0$ itérée 12 fois puis 3 fois 4 fois afin de récupérer les mots intermédiaires comme valeur d'initialisation des deux FCSR. Notons également qu'un key schedule déduit de la fonction *Round* permet d'ajouter une addition de sous-clé à chaque étage.
- **Phase de génération de la suite chiffrante** : les deux FCSR sont clockés en sens opposé, puis le xor de leur valeur $Y(t)$ sert d'entrée à la fonction *Round*. La valeur de

sortie de *Round* $Z(t)$ est mise en mémoire durant 16 itérations. Les bits de suite chiffrante sont alors $s(t) = Y(t) \oplus Z(t-16)$ (dans le cas de X-FCSR-128 la valeur $Y(t)$ est d'abord découpée en deux avant que les deux parties soient xorées entre elles).

1.2.3 La mort de F-FCSR v2 et de X-FCSR v1

A Asiacrypt 2008 [HJ08, HJ11], M. Hell et T. Johansson ont publié une attaque assez dévastatrice contre F-FCSR v2. Cette attaque exploite le fait qu'il existe des corrélations entre les bits de retenues et le bit de rétroaction dans un FCSR en mode Galois. Plus précisément, le contrôle du bit m_0 conduit au contrôle de toutes les rétroactions. Ainsi, si un attaquant est capable de forcer à 0 durant t clocks consécutifs cette valeur, les bits de retenues passent eux aussi à 0 assez rapidement (en raison de la définition même de la fonction de mise à jour des retenues). Si le nombre t de clocks est suffisant, le FCSR devient linéaire conduisant à sa LFSRisation. Plus exactement, son comportement devient affine car il restera toujours un bit de retenue à 1 (comme démontré dans [ABM08]). La probabilité de cet événement dépend de t et elle est approximativement égale à 2^{-t}. Dans ce cas, les bits de sortie et l'état interne LFSRisé sont linéairement dépendants entre eux. L'attaque dans ce cas de figure se transforme en la résolution d'un système linéaire permettant de retrouver le premier état LFSRisé. M. Hell et T. Johansson vont plus loin car en raison de la conduite bijective du FCSR sur son cycle principal, ils peuvent remonter aux états précédents et donc retrouver entièrement la clé. La complexité de l'attaque pour F-FCSR-H v2 est de l'ordre de $\mathcal{O}(2^{25})$ opérations pour un même nombre d'octets de sortie observés (il faut en fait pour permettre la résolution du système linéaire dans le cas de F-FCSR-H v2 que le bit de rétroaction soit nul durant 27 clocks consécutifs).

Dans [SHJ09], les auteurs présentent également une attaque contre X-FCSR-256 utilisant le même principe et permettant de retrouver l'état interne. Dans leur dernier article sur ce sujet [SHJar], ils parviennent à retrouver l'état interne de toutes les versions de X-FCSR v1 en $\mathcal{O}(2^{67})$ pour X-FCSR-128 et en $\mathcal{O}(2^{45})$ pour X-FCSR-256.

Notons également que dans [FMS08], les auteurs proposent une attaque par linéarisation contre un FCSR en mode Fibonacci filtré linéairement. Dans ce cas, les retenues n'influencent qu'un bit du registre principal à chaque clock. Ainsi, un tel générateur sera vulnérable à une attaque où l'adversaire contrôle les bits de retenues et donc une partie du registre principal.

1.2.4 F-FCSR v3 et X-FCSR v2

Afin de contrecarrer ces attaques très efficaces, nous avons cherché une autre manière de représenter les FCSR car les FCSR en mode Galois ou Fibonacci semblaient alors inadéquates pour des applications cryptographiques. Pour cela, nous nous sommes inspirés des travaux faits dans le cas des LSFR [Rog89, JHK06a, MRT04] et du chiffrement par flot Pomaranch [JHK06b]. Ces articles utilisent une représentation matricielle des LFSM (Linear Finite State Machine) pour représenter ce qui est génériquement appelé un ring LFSR. Nous avons donc appliqué ces méthodes afin de construire ce que nous avons nommé des ring FCSR.

ring FCSR

Les FCSR en mode Galois et Fibonacci sont deux automates différents avec des propriétés similaires. Dans un FCSR en mode Galois, la cellule m_0 influence $w_H(d)$ cellules du registre principal tandis que dans un FCSR en mode Fibonacci, la cellule m_{n-1} est modifiée par $w_H(d)$ cellules du registre principal. La représentation ring d'un FCSR est un compromis entre ces deux cas extrêmes :

Définition 1.2.2 *Un ring FCSR est un automate composé d'un registre principal à décalage de n cellules binaires $m = (m_0, \ldots, m_{n-1})$ et d'un registre de retenues (dont les valeurs peuvent être entières) $c = (c_0, \ldots, c_{n-1})$. Il est mis à jour de la manière suivante :*

$$\begin{cases} m(t+1) &= Tm(t) + c(t) \mod 2 \\ c(t+1) &= Tm(t) + c(t) \div 2 \end{cases} \tag{1.1}$$

où $\div 2$ est l'expression $X \div 2 = \frac{X - (X \mod 2)}{2}$ et où T est une matrice $n \times n$ à coefficients 0 ou 1 dans $\mathbb{Z}$, appelée matrice de transition et qui a la forme suivante :

$$\begin{pmatrix} * & 1 & & & & \\ & * & 1 & & (*) & \\ & & * & 1 & & \\ & & & \ddots & \ddots & \\ & & (*) & & * & 1 \\ 1 & & & & & * \end{pmatrix}$$

Dans le cas d'un ring FCSR, toute cellule peut être utilisée comme une rétroaction pour n'importe quelle autre.

Soit $t_{i,j}$ l'élément de la i-ème ligne et de la j-ème colonne de T, on a :

$$T = (t_{i,j})_{0 \le i,j < n} \text{ avec } t_{i,j} = \begin{cases} 1 & \text{si la cellule } m_j \text{ est utilisée pour mettre à jour la cellule } m_i, \\ 0 & \text{sinon.} \end{cases}$$

Comme le registre principal d'un ring FCSR est par définition un registre à décalage, la sur-diagonale de la matrice de transition T est pleine de 1, i.e. pour tout $0 \le i < n$, on a

$$t_{i,i+1 \bmod n} = 1.$$

Par exemple, le ring FCSR avec $q = -347$ présenté à la Fig. 1.5 a pour matrice de transition la matrice de transition T_R présentée à la Fig. 1.6 et le graphe correspondant, à la Fig. 1.6.

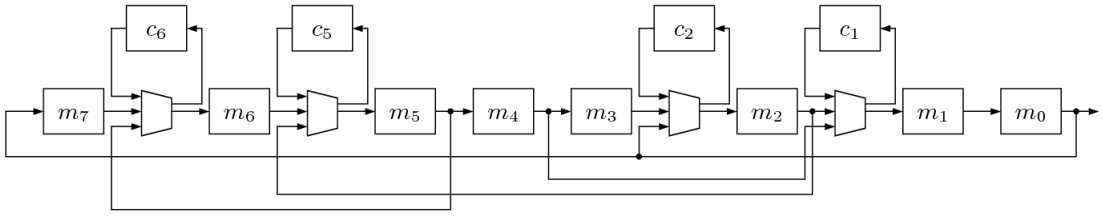


FIGURE 1.5 – Exemple d'un ring FCSR ($q = -347$).

Dans [GK02], M. Goresky et A. Klapper introduisent les représentations pour les FCSR en mode Galois et Fibonacci à l'aide de leur matrice de transition. Ainsi, un FCSR en mode Galois aura une matrice de transition où la première colonne contient les coefficients $d_0, \cdots d_{n-2}$ car dans ce cas, le bit m_0 influence toutes les rétroactions. Pour la matrice de transition d'un FCSR en mode Fibonacci c'est la dernière ligne qui sera composé des éléments $1, d_{n-2}, \cdots d_0$ car toutes les rétroactions influencent une seule cellule m_{n-1}.

De la définition initiale des ring FCSR, on peut déduire beaucoup de propriétés intéressantes, celles-ci sont détaillées dans [ABL+09] (ce dernier article étant celui présenté dans l'Annexe A)

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

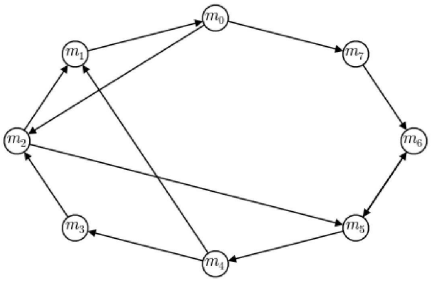


FIGURE 1.6 – Matrice de transition et graphe du ring FCSR de la Fig. 1.5.

et dans [ABP11]. Nous signalerons ici seulement que, une fois la matrice T construite, la valeur de q se déduit en calculant $q = \det(I - 2T)$. De plus, si q est premier et que l'ordre de 2 est maximal dans $\mathbb{Z}/q\mathbb{Z}$ alors chaque série observée dans chacune des cellules du registre principal à partir du temps t est une ℓ-séquence. De plus, les séries produites par deux FCSR avec des matrices de transition différentes mais une unique valeur de q sont équivalentes à la matrice de passage d'une matrice à l'autre près.

Utiliser des ring FCSR pour construire un chiffrement à flot présente de nombreux avantages si sa matrice de transition est bien choisie : son implémentation matérielle est meilleure (voir Annexe A) en terme de chemin critique, de fan-out et de coût. De plus il garantit une diffusion, vue comme le nombre de clocks minimal permettant que toutes les cellules du registre principal soient influencées par au moins une autre, plus rapide.

L'utilisation d'un ring FCSR empêche l'attaque de M. Hell et T. Johansson de se produire. Alors que pour un FCSR en mode Galois, il fallait maîtriser la valeur d'un seul bit de rétroaction durant t clocks, avec un ring FCSR, il faut maîtriser non plus une cellule durant t clocks mais toutes les rétroactions possibles. Si ce nombre est noté k, la probabilité que ces k rétroactions soient nulles durant t clocks devient 2^{-tk}. Avec le ring FCSR utilisé dans F-FCSR-H v3, en regardant 2^{38} états, seuls 41 gardaient des valeurs de retenues constantes durant un clock, aucun durant 2 clocks.

F-FCSR v3

Pour la construction des deux instances de F-FCSR v3 (F-FCSR-H v3 et F-FCSR-16 v3), nous nous sommes d'abord intéressés au choix de la matrice T puis à celui du filtre linéaire et enfin au design du Key/IV setup.

Pour la matrice T, comme nous n'avons pas d'algorithme générique nous permettant, à partir d'un premier q bien choisi, de déterminer la matrice T optimale, nous avons inversé le processus de recherche : c'est-à-dire, à partir de critères que doit vérifier la matrice, nous avons tiré aléatoirement des matrices répondant à ces critères, puis calculé le q correspondant, testé si il était premier et testé si l'ordre de 2 dans $\mathbb{Z}/q\mathbb{Z}$ était maximal. Parmi les critères retenus, on peut citer : un nombre de rétroactions fixé par avance dans un intervalle, une taille de matrice prédéterminée, une bonne implémentation matérielle, etc. Notons également que dans [LP11], les auteurs proposent de construire T à partir de q, il s'agit d'un premier résultat pour cet axe de recherche. Le FCSR finalement choisi comme base de F-FCSR-H v3 a une taille de 160 bits, possède 82 rétroactions et a une diffusion égale à 24.

Pour le choix du filtre linéaire, nous avons retenu deux critères : inclure les cellules qui recevaient une rétroaction afin d'empêcher au maximum toute corrélation et d'utiliser une structure

non périodique. En effet, le filtre linéaire de F-FCSR v2 avait une structure périodique qui permettait d'accélérer l'attaque présentée dans [HJ08].

Le choix du key/IV setup a été plus délicat. En effet, l'attaque de [HJ08] exploitait le fait que le FCSR était dans un état du cycle principal ce qui rendait la recherche d'antécédents et donc de la clé plus simple. Nous avons donc choisi ici de rester le plus longtemps possible dans un état hors du cycle principal pour créer une transformation dure à inverser. Cependant, l'utilisation d'un ring FCSR conduit à une nouvelle difficulté : assurer l'entropie de l'automate. Dans le cas de FCSR en mode Galois ou Fibonacci, annuler le contenu du registre de retenues garantit que nous sommes sur des états non équivalents lors de la convergence sur le cycle principal et donc l'absence de collisions tout en maintenant une entropie constante. Cela vient de la structure de la matrice adjacente $(I-2T)^*$ qui est dans le cas de FCSR en mode Galois ou Fibonacci constituée d'une suite de puissance de 2 sur la première ou la dernière ligne selon le cas. Pour un ring FCSR, il n'existe pas une telle structure. Cependant dans ce cas, la recherche de collisions devient une instance du problème de la somme de sous-ensembles avec une complexité égale à $2^{n/2}$ (si les retenues sont nulles) et égale $2^{3n/2}$ dans le cas général. Ainsi, le nouveau key/IV setup a été construit pour rester le plus longtemps possible sur des états hors du cycle principal tout en limitant la perte d'entropie.

X-FCSR v2

Les ring FCSR définis précédemment ont de bonnes performances matérielles, par contre leurs performances logicielles semblent mauvaises car il n'est pas aussi simple de paralléliser leurs implémentations que dans le cas des LFSR. Nous nous sommes donc intéressés aux méthodes existant dans la littérature pour construire des LFSR rapides en logiciel, l'idée étant de pouvoir travailler sur des mots de taille machine standard (typiquement 8, 16, 32 ou 64 bits) afin d'être au plus près de l'architecture d'un processeur.

Beaucoup de propositions de ce type existaient dans le monde des LFSR. On peut citer ici les travaux de [LP73, Rog89, MK92, MN98, Nie95, Mar03] (voir aussi Section V.C de l'article présenté en Annexe A). Le principe général de toutes ces propositions consiste à construire des matrices de transition pour les LFSR sur des mots de w bits. Ainsi la sur-diagonale remplie de 1 (qui garantit sur $\mathbb{F}_2$ le décalage du registre) devient une sur-diagonale composée de I_w, la matrice identité de taille $w \times w$. Il s'agit ensuite de "compléter" cette matrice à l'aide d'opérations bien choisies (typiquement des décalages et des rotations comme proposés dans [Mar03, PL05a]) sur des mots de taille w bits. Une autre solution, comme proposée pour l'algorithme SNOW v2 [EJ02], est de construire un LFSR sur le corps $\mathbb{F}_{2^w}$.

Nous avons donc dans un premier temps appliqué notre représentation ring aux propositions précédentes utilisant des mots de w bits (ceci est décrit dans la Section suivante) afin d'obtenir des ring LFSR avec de très bonnes propriétés de diffusion et de bonnes performances logicielles. Ensuite, c'est tout naturellement que nous avons dérivé cette définition dans le cas des FCSR (en travaillant sur $\mathbb{Z}_2$) afin d'améliorer leurs performances logicielles et de conserver le caractère non-linéaire de la fonction de mise à jour de l'état.

Comme dans le cas des ring FCSR, ces nouveaux FCSR que nous appellerons word ring FCSR seront complètement déterminés par le choix de la matrice T de transition et la taille w des mots utilisés. Ainsi, un word ring FCSR agissant sur des mots de w bits sera vu comme un registre principal de taille r mots de w bits : $M_0, \cdots, M_{r-1}$ avec r mots de retenue $C_0, \cdots, C_{r-1}$. Sa matrice de transition T de taille $r \times r$ représentera cette nouvelle structure orientée mots de w

bits :

$$\begin{pmatrix} & I_w & & R_0 & & \\ R_1 & & I_w & & (0) & \\ & & & I_w & & R_2 \\ & (0) & & & \ddots & \\ & & R_{r-2} & & & I_w \\ I_w & R_{r-1} & & & & \end{pmatrix}$$

où I_w est la matrice identité de taille binaire $w \times w$ et où les R_i représentent des opérations orientées mots de w bits. Les R_i peuvent être des décalages ou des rotations garantissant une certaine efficacité logicielle et qui sont les opérations les plus simples pouvant être utilisées (voir [Mar03, PL05a]).

Afin de présenter un exemple, on introduit les notations matricielles suivantes sur un mot de w bits :

$$\begin{cases} SL \cdot (x_0, \cdots, x_{w-1})^t = (x_1, \cdots, x_{w-1}, 0)^t \\ SR \cdot (x_0, \cdots, x_{w-1})^t = (0, x_0, x_1, \cdots, x_{w-2})^t \end{cases}$$

où SL est l'opération matricielle sur un mot de w bits représentant un décalage à gauche et où SR représente un décalage à droite. Les rotations à gauche (RL) et à droite RR sont définies de la manière suivante :

$$\begin{cases} RL \cdot (x_0, \cdots, x_{k-1})^t = (x_{k-1}, x_0, x_1, \cdots, x_{k-2})^t \\ RR \cdot (x_0, \cdots, x_{k-1})^t = (x_1, \cdots, x_{k-1}, x_0)^t \end{cases}$$

En utilisant ces notations, les paramètres R_i de la matrice T précédente sont égaux à SL^a, SR^b, RL^c ou RR^d où a, b, c et d désignent les valeurs de décalage. Ainsi, la matrice T suivante définit un word ring FCSR sur des mots de $w = 8$ bits :

$$T = \begin{pmatrix} 0 & I & 0 & SL^3 & 0 \\ 0 & 0 & I & 0 & SR^2 \\ 0 & 0 & SL^1 & I & 0 \\ SR^3 & 0 & 0 & 0 & I \\ I & 0 & 0 & 0 & 0 \end{pmatrix}$$

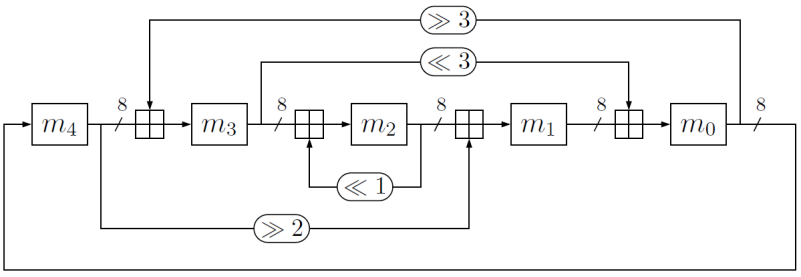


FIGURE 1.7 – Un exemple de word ring FCSR

Le FCSR associé est donné à la Fig. 1.7 avec $n = 40$ et $w = 8$. La valeur de q correspondante peut être complètement déterminée en utilisant la formule $q = \det(I - 2T)$ et vaut -1497813390989. Ce nombre étant premier et d'ordre maximal, le FCSR ainsi défini produit bien une ℓ-séquence, il est efficace en logiciel mais également en matériel (il requiert seulement 24 additionneurs binaires).

Notons également que cette définition de word ring FCSR permet au même titre que les ring FCSR d'empêcher l'attaque de M. Hell et T. Johansson par LFSRisation.

A Indocrypt 2009 [BMP09], nous avons donc proposé une nouvelle version de la famille X-FCSR (en gardant exactement la même méthode d'extraction avec la fonction *Round*, les 16 registres de mémoire et le Key/IV setup) utilisant un unique word ring FCSR de 512 bits agissant sur des mots de 32 bits. Nous avons également proposé une version de F-FCSR, F-FCSR-32, dédiée au logiciel qui utilise le même word ring FCSR de 512 bits avec des mots de 32 bits filtré par une fonction linéaire extrayant 32 bits à chaque clock.

1.3 Quelques résultats concernant les LFSM et les LFSR

Parallèlement à cette étude sur les FCSR, nous nous sommes également rendu compte qu'une étude générale de la même ampleur méritait d'être faite pour les LFSM et les LFSR (B. Pousse durant sa thèse [Pou10] a également réalisé le même type de généralisation au cas des AFSR). L'article résumant tous ces travaux [ABMPar] est donné en Annexe A. En effet, même si les résultats concernant la représentation matricielle des LFSM et des LFSR étaient connus depuis longtemps [Gol81], l'impact de ceux-ci sur les performances hardware et software des LFSR n'était pas clair ; la notion de diffusion restait extrêmement liée aux critères de corrélation, d'auto-corrélation et de cross-corrélation [GG04] et n'existait pas en tant que telle ; les différentes variantes des word ring LFSR méritaient une définition unificatrice et plus générale. Nous avons également introduit dans cet article une nouvelle représentation polynomiale des LFSM, c'est-à-dire que les coefficients de la matrice de transition T sont des quotients de polynômes. Cette représentation a un intérêt direct permettant une description simple des "générateurs windmill" introduits par Smeets et Chambers [SC88].

Je ne détaillerai pas ici les résultats équivalents pour les LFSR à ceux introduits pour les FCSR. Ils se déduisent directement de la partie précédente et sont donnés en Annexe A. Je n'introduirai ici que la nouvelle représentation polynomiale et donnerai un exemple de son intérêt.

1.3.1 Notations et définitions

On note $\mathbb{F}_q$ le corps fini de cardinal q, $\mathbb{F}_q[X]$ désignera l'anneau des polynômes sur $\mathbb{F}_q$ et $\mathbb{F}_q[[X]]$ l'anneau des série entières sur $\mathbb{F}_q$. L'anneau $\mathcal{Q}$ désignera l'anneau des séries entières rationnelles, c'est-à-dire celles qui peuvent être écrites $P(X)/Q(X)$ avec $P, Q \in \mathbb{F}_q[X]$ et $Q(0) \neq 0$. $\mathcal{Q}$ est donc l'anneau des séries entières qui sont des séquences ultimement périodiques.

Soit A une matrice définie sur $\mathcal{Q}$. On note $A_{i,j}$ son coefficient à la ligne i et à la colonne j : $A_{i,j} = P_{i,j}(X)/Q_{i,j}(X)$. Pour un i fixé, on note : $Q_i(X) = \text{ppcm}(Q_{i,1}(X), \ldots, Q_{i,n}(X))$. Ainsi, on peut réécrire chaque $A_{i,j}$ comme $A_{i,j} = R_{i,j}(X)/Q_i(X)$. Pour chaque ligne i, on définit le sous-ensemble suivant de $\mathcal{Q}$: $W_i = \{R(X)/Q_i(X) \mid \deg(R(X)) < \max_j(\deg(R_{i,j}(X)))\}$. On définit finalement W comme $W = \prod_{i=1}^{n} W_i \subset \mathcal{Q}^n$. W est un ensemble fini.

Définition 1.3.1 *Une Machine Linéaire à États Finis Rationnelle (en anglais Rational Linear Finite State Machine) RLFSM avec l bits de sorties et de longueur n sur $\mathcal{Q}$ est un automate à états finis défini par une paire (A, C) de matrices à coefficients dans $\mathcal{Q}$ de tailles respectives $n \times n$ et $l \times n$. L'ensemble des états de cet automate est $\mathbb{F}_2^n \times W$ où W est défini à partir de A comme vu précédemment. En considérant que l'automate est dans l'état $(m^{(t)}, c^{(t)})$ au temps t et que $v^{(t)}$*

est la sortie au temps t, les fonctions de transition et d'extraction au temps t sont définies par :

$$\begin{cases} m^{(t+1)} &= Am^{(t)} + c^{(t)} \bmod X \\ c^{(t+1)} &= Am^{(t)} + c^{(t)} \operatorname{div} X \\ v^{(t)} &= Cm^{(t)} \end{cases}$$

On en déduit la proposition suivante (la preuve est donnée en Annexe A)

Proposition 1.3.2 *Soit $\mathcal{L}$ une RLFSM définie par sa matrice de transition A et sa matrice de sortie C. Soit $T(X) = \det(G(X)(I - XA))$. Les séquences de sortie $V_i^{(t)}$ sont des séries entières rationnelles de la forme $P_i(X)/T(X)$.*

1.3.2 Exemple : le générateur windmill de E0

Les générateurs windmill peuvent être vus comme des LFSM sans entrée et avec plusieurs sorties. Ils ont été introduits dans [SC88] comme une LFSM composée de v ($v \geq 1$) LFSM connectées en cascade pour former un cycle. Chacune de ces LFSM est appelée une pale du windmill. Classiquement, on utilise la représentation en mode Fibonacci pour chacune de ces LFSM. Cependant, on peut montrer que la représentation avec des LFSM en mode Galois est équivalente. C'est ce que nous ferons dans cet exemple. Les générateurs windmill sont caractérisés par leurs connexions avants et arrières. Ces connexions sont identiques pour toutes les pales même si la longueur des LFSM peut être différente car elles peuvent être décalées dans différentes LFSM.

Les générateurs windmill permettent d'accélérer la production de t bits issue d'un LFSR. Au lieu de générer un bit après l'autre, on décompose la séquence en t sous-séquences et à chaque cycle, un bit de chaque sous-séquence est produit [Gün89]. Ainsi, si on considère une séquence $S = (s_n)_{n\in\mathbb{N}}$, un générateur classique sortira s_0 au premier clock, s_1 au deuxième, etc. tandis qu'un automate parallèle sera capable de sortir v bits à chaque clock : $(s_0, s_1, \ldots, s_{v-1})$ au premier clock, $(s_v, \ldots, s_{2v-1})$ au second, etc. Plus précisément, un automate parallèle a v sorties qui produisent les séquences $S^i := (s_{nv+i})_{n\in\mathbb{N}}$ avec $0 \leq i < v$. Par la suite, on s'intéressera à la caractérisation des séquences S^i et non à la séquence reconstruite S.

Le générateur windmill présenté à la Fig. 1.8 est celui utilisé dans le chiffrement à flot E0 [Blu01] à la représentation en mode Galois/Fibonacci près. Il est constitué d'une pale de longueur 7 et de trois pales identiques de longueur 6. Aucune rétroaction n'apparaît, seules des connections avant sont présentes sur les cellules (par exemple m_{13} est connectée aux cellules m_{12}, m_{10}, m_9 et m_7).

Jusqu'à présent, seuls les générateurs windmill avec une seule pale répétée plusieurs fois ont été étudiés. Cette définition peut être généralisée afin d'autoriser l'utilisation de pales différentes pour un même générateur windmill. Avec l'exemple précédent, on peut considérer les séquences de sortie des cellules m_0, m_7, m_{13} et m_{19} et représenter chaque pale par un polynôme. Cela conduit à l'interprétation présentée à la Fig. 1.9.

La LFSM décrite à la Fig. 1.9 a donc la matrice de transition en représentation rationnelle suivante :

$$(X^5 + X^3 + X^2 + 1) \cdot \begin{pmatrix} 0 & X & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

En suivant la définition 1.3.1, les générateurs windmill comme introduits par Smeets et Chambers [SC88] peuvent être définis comme :

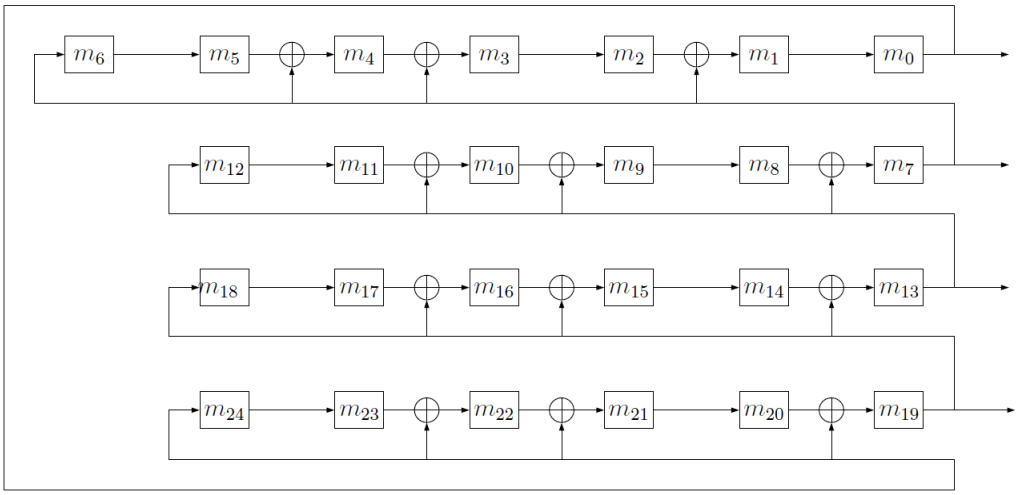


FIGURE 1.8 – Le générateur windmill de E0

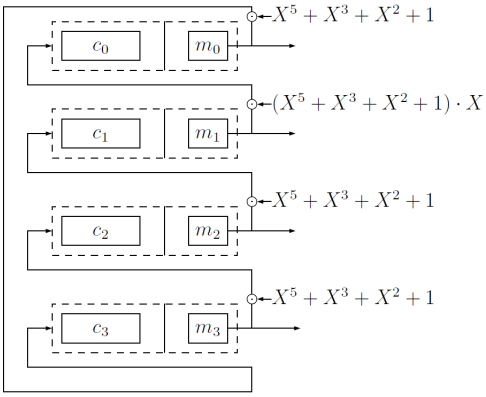


FIGURE 1.9 – Un générateur windmill en représentation rationnelle.

Définition 1.3.3 *Un générateur windmill défini par les polynômes $\alpha(X), \beta(X)$ avec $\beta(0) \neq 0$ et v vanes est un LFSR de longueur v avec comme matrice de transition A sur $\mathbb{F}_2[[X]]$:*

$$\begin{pmatrix} 0 & \frac{\alpha(X)}{\beta(X)}X^{i_0} & & (0) \\ \vdots & \ddots & \ddots & \\ 0 & (0) & \ddots & \frac{\alpha(X)}{\beta(X)}X^{i_{v-2}} \\ \frac{\alpha(X)}{\beta(X)}X^{i_{v-1}} & 0 & \dots & 0 \end{pmatrix}$$

où $0 \leq i_0, \dots, i_{v-1}$.

Avec cette définition, chaque ligne représente une pale du générateur windmill. En particulier, la longueur de la pale j est $\max(\deg(\alpha(X)X^{i_j}), \deg(\beta(X)))$.

On peut également directement calculer $\det(I - XA)$: $\det(I - XA) = X^n\,(\alpha(X)/\beta(X))^v + 1$, où $n = i_0 + \cdots + i_{v-1}$. On pose $Q(X) = X^n\alpha(X)^v + \beta(X)^v$, on obtient $\det(I - XA) = Q(X)/\beta(X)^v$. Les séquences $M_i^{(t)}$ observées à la sortie de cette RLFSM sont de la forme $P_i(X)/Q(X)$. Le résultat principal sur les générateurs windmill (voir [SC88]) est l'existence d'une permutation

σ de $\{0, \ldots, v-1\}$ telle que la série $S(X) = \sum_t (\sum_{i=0}^{v-1} m_i(t) X^{\sigma(i)}) X^{vt}$ est une série entière rationnelle de la forme $P(X)/Q(X^v)$. En d'autres termes, un générateur windmill est capable de sortir en parallèle à chaque itération v valeurs consécutives d'une série entière rationnelle, le cas le plus intéressant étant bien évidemment celui où $Q(X^v)$ est un polynôme primitif.

L'approche polynomiale présentée succinctement ici est détaillée dans l'article [ABMPar] et donnée dans l'Annexe A. Elle donne une caractérisation simple et unique des générateurs windmill. En particulier, cette approche montre que les propriétés des générateurs windmill (la génération parallèle d'une m-séquence donnée) sont indépendantes de l'implémentation de chaque pale. Ainsi, on peut envisager cette implémentation avec des pales en mode Fibonacci ou en mode Galois, ou plus généralement avec des pales en ring, ce qui permettra une meilleure diffusion et une meilleure implémentation.

Chapitre 2

Algorithmes de chiffrement par blocs et fonctions de hachage

2.1 Introduction

Il peut sembler étrange de rassembler dans un même chapitre des résultats concernant à la fois les chiffrements par blocs et les fonctions de hachage, ces deux primitives n'ayant pas du tout les mêmes buts cryptographiques. Cependant, d'un point de vue cryptanalytique et au vu des dernières attaques contre les chiffrements par blocs, cette catégorisation peut faire sens par effet boomerang. En effet, au début des années 2000, ce sont les techniques de cryptanalyses de chiffrements par blocs, comme la cryptanalyse différentielle, qui ont été appliquées pour la recherche de collisions aux fonctions de hachage (voir par exemple les attaques [WYY05b, WYY05a] contre la famille des SHA). Aujourd'hui, ce sont les résultats de cryptanalyses des fonctions de hachage qui servent à attaquer des chiffrements par blocs dans des modèles à clés connues ou même à clés choisies. Nous reviendrons sur ces notions tout au long de ce chapitre. De même, si on ose occulter l'actuelle compétition SHA-3 [Nat07], les nouveaux chiffrements par blocs et les nouvelles fonctions de hachage proposés cherchent à atteindre un même but : être efficace en matériel même si les méthodes utilisées pour atteindre cet objectif diffèrent.

Chiffrements par blocs

De façon générique, on décrit un algorithme de chiffrement par blocs E comme une application de l'espace des messages $\mathcal{M}$ et de l'espace des clés $\mathcal{K}$ vers l'espace des textes chiffrés $\mathcal{C}$ où la relation $E_K(m) = c$ (avec $m \in \mathcal{M}$, $K \in \mathcal{K}$ et $c \in \mathcal{C}$) est vérifiée pour l'ensemble des éléments des différents ensembles. Donc E_K doit être un ensemble de permutations sur l'ensemble de toutes les clés possibles. De plus, l'algorithme de déchiffrement correspondant D_K qui permet de déchiffrer c avec la même clé K doit vérifier : $D_K(c) = m$ c'est-à-dire $\forall m,\ D_K(E_K(m)) = m$ autrement dit $D_K = E_K^{-1}$ pour toute clé K dans $\mathcal{K}$. La permutation de chiffrement E_K paramétrée par la clé K doit également être facile à inverser. De plus, la clé K doit être prise dans un espace suffisamment grand pour se prémunir contre la recherche exhaustive (essayer toutes les clés). Par exemple dans le cas de l'AES, K est de taille 128, 192 ou 256 bits, ce qui fait que la recherche exhaustive nécessite 2^{128}, 2^{192} ou 2^{256} essais, cette dernière borne étant largement supérieure au nombre d'atomes présents dans l'univers. De même, la taille des espaces $\mathcal{M}$ et $\mathcal{C}$ (qui sont en général les mêmes) doit être suffisamment grande pour éviter les attaques par dictionnaire. Notons également que comme son nom l'indique, un chiffrement par blocs prendra en entrée/sortie des

blocs de taille n bits avec typiquement $n = 64$ ou 128.

Évidemment, même si la taille des espaces considérés est une donnée essentielle à prendre en compte, cela ne suffit pas à garantir la sécurité d'un algorithme de chiffrement par blocs. C'est pourquoi une partie de la recherche théorique du domaine s'est attachée à définir des modèles de sécurité dépendant de la puissance de l'attaquant (peut-il choisir les textes claires seulement ou également les textes chiffrés, a-t'il accès à E_K uniquement ou à E_K^{-1} ?) et à prouver la résistance d'un chiffrement E_K (ou d'une structure de chiffrement) en montrant qu'il est impossible de le distinguer d'un chiffrement idéal (c'est-à-dire d'une permutation aléatoire parfaite C^*) en considérant que le nombre d'essais auxquels a droit l'attaquant est borné. Ainsi, on obtient des bornes inférieures sur la probabilité de distinguer, pour un nombre de clairs/chiffrés bornés et une complexité donnée, un chiffrement d'un chiffrement idéal permettant de garantir la sécurité de la construction.

Lorsque l'on souhaite chiffrer un message m de taille n bits à l'aide d'un algorithme de chiffrement par blocs, on utilise des modes de chiffrements qui permettent d'enchaîner le chiffrement de plusieurs blocs de taille n bits après avoir découpé le message m en blocs m_i. Parmi les modes de chiffrement les plus connus, on peut citer les modes ECB (Electronic Code Book), CBC (Cipher Block Chaining) ou CTR (Counter).

Les méthodes de construction de la primitive E_K restent très empiriques mais la plus usitée est l'utilisation d'algorithmes itératifs. Ces algorithmes sont composés de r étages et à chaque étage, ils font agir la même fonction f paramétrée par une sous-clé K_i. Les sous-clés $K_1, \cdots, K_r$ sont différentes à chaque étage et générées à partir de la clé secrète K et d'un algorithme de génération de sous-clés.

Notons que, mis à part quelques cas pathologiques, plus le nombre d'étages est grand, plus on augmente la diffusion (faire en sorte que chaque bit de sortie soit influencé par le plus grand nombre possible de bits d'entrée) et la confusion (faire disparaître les structures tant linéaires qu'algébriques du chiffrement) selon les principes décrits par Shannon dans [Sha49].

Dans ce cas, le principal défi reste la construction de la fonction d'étage f. Il existe essentiellement deux méthodes pour construire la fonction f :

- **Schéma de Feistel :** Si f_1 est une fonction de $I_n = \{0,1\}^n$ dans lui-même et x^0, x^1, x^2, x^3 quatre éléments de I_n. On définit le schéma de Feistel Ψ lié à f_1 sur $I_{2n} = \{0,1\}^{2n}$ de la manière suivante : $\forall (x^0, x^1) \in (I_n)^2, \Psi(f_1)[(x^0, x^1)] = (x^2, x^3)$ avec $x^2 = x^1$ et $x^3 = f_1(x^1) \oplus x^0$. Pour toute fonction f_1, $\Psi(f_1)$ est une permutation de I_{2n}, plus précisément, c'est une involution à une permutation des deux entrés près. Cette définition permet de construire un étage d'un chiffrement par blocs itératif, le processus étant itéré sur r étages pour générer le chiffrement en entier. Le chiffrement par blocs le plus célèbre utilisant un schéma de Feistel est bien évidemment le DES [Nat77]. Il est également possible de modifier ce schéma originel en multipliant le nombre de branches d'entrées/sorties (c'est le cas du schéma utilisé dans MARS [CBD+98], candidat malheureux de la compétition AES (1997-2001) [oST01]) ou en changeant la position de la fonction f_1 (c'est le cas du schéma utilisé dans MISTY [Mat97] et KASUMI [Pro01], appelé schéma L).
- **Réseau de Substitution/Permutation (S/P) :** Un réseau de Substitution/Permutation utilisé comme fonction d'étage d'un chiffrement par blocs découpe tout d'abord le bloc courant de n bits en m sous-blocs de taille k bits. Puis, il applique une boîte S (permutation non-linéaire choisie pour ses bonnes propriétés de résistance aux cryptanalyses connues et de confusion [Sha49]) à chacun des sous-blocs ; ensuite, il applique au bloc de taille n une transformation (en général linéaire) choisie pour ses bonnes propriétés de diffusion [Sha49] ; finalement, un ou-exclusif est appliqué entre chaque bit de l'état courant et chaque bit de la sous-clé de l'étage. L'AES [FIP01] et la famille de chiffrement SAFER [Mas93, MKK00]

sont fondés sur des réseaux S/P.

Fonctions de hachage

Les fonctions de hachage sont les outils cryptographiques les plus utilisés au monde. En cryptographie, une fonction de hachage H prend en entrée un message de taille variable et produit une empreinte de ce message de taille fixe, n bits (typiquement 224, 256 ou 512 bits). Cette fonction doit vérifier les trois propriétés suivantes :

- **Preimage résistante :** étant donnée une sortie y, il est difficile de trouver x tel que $H(x) = y$. Cela signifie que H est à sens unique.
- **Seconde preimage résistante :** étant donnée une entrée x, il est difficile de trouver x' tel que $H(x') = H(x)$. Cela signifie que H est faiblement résistante aux collisions.
- **Résistante aux collisions :** Il est difficile de trouver x et x' tels que $H(x') = H(x)$. Cela signifie que H est fortement résistante aux collisions.

Ces définitions restent vagues mais peuvent quand même être quantifiées dans le cas qui nous intéressera par la suite à savoir les fonctions de hachage itératives. Une fonction de hachage itérative est composée d'une part d'une fonction de compression qui prend en entrée une valeur de chaînage H_i de taille fixe et un bloc de message à hacher M_i également de taille fixe et qui produit la valeur de chaînage suivante H_{i+1}. Dans ce cas, le message d'origine M à hacher est découpé en t blocs M_i où chaque bloc représentera une entrée de la fonction de compression. D'autre part, cette fonction de hachage fait appel à un mode opératoire qui décrit comment itérer la fonction de compression et ce jusqu'à épuisement des blocs M_i.

Si la taille de la valeur de chaînage H_i est n bits, la résistance stricte aux collisions implique qu'il n'existe pas d'algorithme permettant de trouver une collision sur H avec une complexité inférieure à $2^{n/2}$ appels à la fonction de compression (borne des anniversaires). Dans ce cas, la (seconde) preimage résistance de H au sens strict implique qu'il n'existe pas d'algorithme permettant de trouver une (seconde) preimage sur H dont la complexité moyenne est inférieure à 2^n appels à la fonction de compression.

En ce qui concerne les modes opératoires, le plus célèbre reste la construction de Merkle-Damgård [Dam89, Mer89] : H_0 est initialisé à l'aide d'une valeur d'initialisation publique IV et $H_{i+1} = f(H_i||M_i)$, la sortie de la fonction de hachage étant la dernière valeur H_{t+1}. La manière de faire intervenir le bloc précédent H_i dans la fonction de compression varie selon le mode utilisé. On peut par exemple citer les modes Matyas-Meyer-Oseas (MMO) [MMO85] ou Miyaguchi-Preneel. Ces modes peuvent notamment utiliser un chiffrement par blocs comme fonction de compression, la clé utilisée dans ce cas est alors remplacée par la valeur H_i.

Historiquement, les deux grandes familles de fonctions de hachage sont les MD (MD4 [Riv90] et MD5 [Riv92b]) et les SHA (pour Secure Hash Algorithm, avec ses variantes : SHA-0/1/2 [oC93, oC95]) et utilisent des fonctions de compression itératives, fondées sur des variantes d'une même fonction de tour. En 2004 et 2005, des attaques dévastatrices ont été publiées contre MD4, MD5, SHA-0 et SHA-1 [WY05, WYY05b, WYY05a], les rendant vulnérables voire inutilisables. Même si SHA-2 semble encore épargné, sa sécurité ne fait plus l'unanimité.

C'est pourquoi en 2007, le NIST (National Institute for Standards and Technology) a lancé un appel à primitives pour remplacer les fonctions de hachage existantes : SHA-3 [Nat07]. En Octobre 2008, ce ne sont pas moins de 64 propositions qui ont été soumises, parmi lesquelles 56 ont été jugées comme répondant au cahier des charges du NIST et sont donc entrées en phase 1. Parmi ces propositions, en Juillet 2009, 14 finalistes ont été retenues pour la phase 2 et finalement en Décembre 2010, 5 finalistes ont été gardées parmi lesquelles sera choisie une unique vainqueuse en 2012 qui deviendra par la suite le standard pour le hachage (pour plus

de détails sur la compétition et toutes les primitives proposées, le lecteur peut consulter le site SHA-3 Zoo [7]).

Dans les nouvelles méthodes de construction proposées durant la compétition SHA-3, on peut citer les modes particuliers de Shabal [BCCM+08], l'utilisation de fonctions éponge [BDPA08] ou les variantes de HAIFA [BD07]. En ce qui concerne les fonctions de compression elles-mêmes, beaucoup comme ECHO [BBG+09], Grøstl [GKM+11], LANE [Ind08] ou SHAvite-3 [BD09] utilisent des briques de base de l'AES.

La compétition SHA-3 concentre une grande partie de l'attention cryptographique actuelle et permet une avancée concernant les connaissances sur les fonctions de hachage. Il faudra malheureusement attendre encore un peu avant de connaître le standard.

Notions de distingueurs

Nous donnerons tout d'abord ici la définition d'un distingueur puis nous préciserons dans quel cadre cette notion peut être utilisée.

Un distingueur est une machine de Turing probabiliste qui cherche à distinguer par un jeu de questions/réponses une famille de fonctions f_K d'une fonction de chiffrement idéale f^* :

Définition 2.1.1 *Soit deux ensembles M_1 et M_2 et une famille de fonctions f_K de M_1 vers M_2 paramétrées par un secret K tiré aléatoirement dans l'ensemble $\mathcal{K}$. Un distingueur pour la famille de fonctions f_K est une machine de Turing probabiliste $\mathcal{A}$ disposant d'un oracle $\mathcal{O}$ qui pour toute requête de M_1 envoie une réponse dans M_2. Après un certain nombre de calculs de l'oracle, le distingueur fournit la réponse "0" ou "1". On caractérise un distingueur par le nombre de requêtes q de l'oracle et le temps de calcul T. On mesure alors l'avantage de $\mathcal{A}$ à distinguer f_K de f^*, fonction aléatoire parfaite (i.e. tirée au sort selon la loi uniforme dans l'ensemble des fonctions de M_1 dans M_2), en calculant $Adv_{f_K}(\mathcal{A}) = |p - p^*|$ où p (respectivement p^*) représente la probabilité que $\mathcal{A}$ réponde 1 lorsque la fonction $\mathcal{O} = f_K$ est implémentée (respectivement $\mathcal{O} = f^*$).*

On peut également utiliser les distingueurs dans le cas de permutations aléatoires, on prend alors en compte les probabilités liées à ces permutations. Il existe deux types de distingueurs : les distingueurs dits "non-adaptatifs" qui préparent à l'avance toutes les requêtes et les distingueurs dits "adaptatifs" qui adaptent les requêtes en fonction des réponses précédentes, cette dernière notion étant bien évidemment plus forte. Dans ces deux cas, les distingueurs peuvent tester des fonctions pseudo-aléatoires (PRF pour Pseudo-Random Function) ou des permutations pseudo-aléatoires (PRP pour Pseudo-Ramdom Permutation).

Il existe également une classe plus forte de distingueurs appelés distingueurs super-pseudo-aléatoires (SPRP pour Super Pseudo-Random Permutation) où l'oracle peut faire appel au chiffrement C ou à son inverse C^{-1}. Dans ce cas, l'utilisation d'un distingueur super-pseudo-aléatoire n'a de sens que dans le cas de l'étude de permutations.

Cette notion de distingueur est utile d'une part pour le cryptanalyste qui va chercher à prouver l'existence d'un distingueur pour une relation particulière existant sur un certain nombre de tours de la fonction considérée avec un avantage le plus grand possible, c'est-à-dire que l'on cherche des relations qui ont une probabilité de se produire la plus éloignée possible de la probabilité uniforme. Cette notion en cryptanalyse peut donc s'appliquer aux attaques contre un chiffrement par blocs afin de déterminer par un test d'hypothèse la bonne clé, celle qui produit le biais le plus proche du biais impliqué par la relation dans le distingueur. Cette notion a également

7. `http://ehash.iaik.tugraz.at/wiki/The_SHA-3_Zoo`

beaucoup été utilisée pour les fonctions de hachage lors de la compétition SHA-3 [8] pour les deux raisons suivantes : dans beaucoup de protocoles, les fonctions de hachage servent à simuler un oracle aléatoire [BR93], il faut donc que ces fonctions soient capables de produire un aléa de très bonne qualité ; de plus, l'existence d'un distingueur pour une relation particulière est souvent le premier pas vers une attaque plus puissante comme une attaque par collision ou une attaque par preimage.

D'autre part, la notion de distingueur sert à construire des preuves de sécurité pour les algorithmes de chiffrement par blocs comme initialement proposé par Luby et Rackoff dans [LR88] qui ont développé l'étude théorique d'un schéma de Feistel à trois étages : ils ont comparé une fonction idéale à un schéma composé de fonctions idéales afin d'évaluer la qualité des schémas utilisés en chiffrement. Ils prouvent la sécurité de ce schéma en comparant les distributions de probabilités entre la fonction de chiffrement engendrée par ce schéma et une fonction idéale. Ils démontrent ainsi une borne supérieure de sécurité sur l'avantage du distingueur obtenu. En d'autres termes, ils prouvent qu'un schéma particulier composé de fonctions idéales est bien une PRP.

La définition de distingueur donnée ici s'applique de façon directe au cas précédent. Par contre, lorsque l'on cherche à élargir cette définition aux fonctions de compression ou à des propriétés à clés connues, de nombreuses difficultés se présentent. Tout d'abord parce que le distingueur défini ici ne peut être utilisé dans le modèle du chiffrement idéal où on cherche à distinguer une famille de permutations aléatoires indexée par une donnée publique (voir par exemple [CPS08]). De plus, il est impossible de définir un distingueur pour une fonction complètement spécifiée (voir [CGH04]) car il ne s'agit plus de s'intéresser à une famille de fonctions mais à une instance particulière. Ainsi, l'unification des différentes notions de distingueurs n'est pas un problème simple en cryptographie symétrique car toutes les modélisations existantes dépendent fortement de la connaissance complète ou non de la famille de fonctions à distinguer. Le modèle proposé ici se limite clairement au cas où le paramètre de la famille de fonctions est secret, nous verrons cependant dans la suite de ce manuscrit comment l'étendre au cas où la clé est connue.

2.2 Chiffrement par blocs : nouveaux paradigmes pour la cryptanalyse

Le but premier de la cryptanalyse contre les algorithmes de chiffrement par blocs est de trouver des techniques permettant de retrouver tout ou partie de la clé utilisée plus rapidement que la recherche exhaustive en utilisant des propriétés structurelles ou statistiques vraies pour toute clé qui ne doivent pas se produire dans le cas de permutations aléatoires. Cela conduit à la construction de distingueurs particuliers.

2.2.1 Classification des distingueurs

Cette direction de recherche a récemment connu un tournant particulier avec l'unification des techniques de cryptanalyses entre les chiffrements par blocs et les fonctions de hachage car aujourd'hui, on considère que les chiffrements par bloc peuvent être utilisés comme des fonctions de hachage grâce à l'utilisation de modes opératoires appropriés et inversement. Auparavant, seules les techniques différentielles étaient appliquées à la fois aux chiffrements par blocs et aux fonctions de hachage. Aujourd'hui les attaques par différentielle impossible [Knu99, BBS99b], les méthodes boomerang [Wag99, KKS01] ou slide [BW99] et les distingueurs dans les modèles

8. dans ce cas, on considère que le secret K tiré aléatoirement est généralement le message M.

à clés connues ou à clés inconnues sont appliquées indifféremment aux deux types de primitive. Les distingueurs à clés connues contre les chiffrements par blocs ont été introduits par Knudsen et Rijmen en 2007 dans [KR07], ceux à clés choisies par Biryukov et al. en 2009 dans [BKN09]. Ces nouveaux types de distingueurs ont permis d'attaquer des versions complètes de l'AES-192 et de l'AES-256 [BKN09, BK09]. Concernant l'AES-128, Mendel et al. [MPRS09] ont présenté à SAC 2009, une attaque à clés connues contre 7 tours de l'AES-128 avec une meilleure complexité que celle initialement proposée dans [KR07] en utilisant la technique dite par rebond [MRST09]. Dans [GP10], H. Gilbert et T. Peyrin ont amélioré ce distingueur en ajoutant un tour grâce à une technique appelée attaque par Super-boîte S. Ces deux dernières techniques seront décrites succinctement dans la Section 2.3.1.

Par exemple, dans le modèle à clés connues, les distingueurs par différentielle tronquée permettent de construire des presque-collisions dans des fonctions de compression fondées sur des chiffrements par blocs. En effet, si on utilise le mode Matyas-Meyer-Oseas, en notant $E_K(P)$ un chiffrement par blocs défini à l'aide d'une clé K et d'un texte clair P, H_{i-1} la variable de chaînage d'entrée et M_i le bloc de message en entrée de la fonction de compression f produisant en sortie la variable de chaînage H_i : $H_i = f(H_{i-1}, M_i) = E_{H_{i-1}}(M_i) \oplus M_i$, une presque-collision entre deux textes chiffrés $C = E_K(P)$ et $C' = E_K(P')$ se transforme en une presque-collision sur les variables de chaînage H_i et H'_i, sorties de la fonction de compression utilisée en mode MMO.

Nous proposons donc la classification suivante pour tenter de capturer ces nouvelles notions :

- modèle *unknown key (UK)* : c'est le cas classique du distingueur où la clé est inconnue et où l'adversaire a accès à une boîte noire pour chiffrer ou déchiffrer les requêtes à l'oracle. Ce modèle inclut bien sûr la possibilité de faire des requêtes à textes connus, à textes clairs choisis, à textes chiffrés connus ou à textes chiffrés choisis. Le but de l'adversaire peut être dans ce cas de retrouver la clé inconnue.
- modèle *unknown related-key (URK)* [Bih93] : une variante du modèle classique de l'adversaire a été proposée par E. Biham en 1993 où l'adversaire a accès à plusieurs oracles qui utilisent des clés différentes mais où il existe des relations connues ou choisies entre les sous-ensembles de ces clés, la clé à retrouver étant toujours inconnue.
- modèle *known key (KK)* [KR07] : celui-ci a été proposé par Knudsen et Rijmen et on suppose ici que la clé utilisée pour le chiffrement est connue même si sa valeur n'est pas contrôlée par l'adversaire.
- modèle *chosen key (CK)* [BKN09] : un adversaire encore plus fort est celui qui est capable de choisir la valeur de la clé utilisée pour le chiffrement.

Même si les capacités de l'adversaire diffèrent en fonction du modèle proposé, tous les modèles partagent un but commun : résoudre le problème du distingueur, à savoir l'adversaire cherche à construire une propriété qui permet de distinguer un chiffrement par blocs avec clé(s) d'une permutation aléatoire.

Nous avons introduit dans [MP] une notion intermédiaire entre les modèles KK et CK, il s'agit du modèle dit à *Related Known-Key (RKK)* où, comme dans le modèle KK l'adversaire construit un ensemble de clés liées entre elles et connaît la clé. La puissance de l'attaquant est bien sûr plus forte que le modèle KK mais plus faible que le modèle CK.

2.2.2 Quelques distingueurs classiques dans le modèle UK

Exemple du distingueur générique non-adaptatif limité à n requêtes

Afin de spécifier ce distingueur (voir Alg. 1), nous utilisons la définition donnée dans [Jun03]. Nous suivrons cette formalisation utilisée par S. Vaudenay dans la théorie de la décorrelation

dans tout le reste de cette section. On donne à l'algorithme 1 un oracle $\mathcal{O}$ qui implémente avec la probabilité 1/2 soit E_K soit G, une permutation aléatoire. Le cœur du distingueur est sa région d'approbation $A^{(n)}$: elle définit l'ensemble des valeurs d'entrée $\mathbf{P} = (P_1, \cdots, P_n)$ qui conduisent à la réponse 1 (l'oracle implémente E_K) ou 0 (l'oracle implémente G). Si une relation particulière R qui définit la région d'approbation existe, reliant ensemble les entrées et les sorties $(R(\mathbf{P}, \mathbf{C}))$ pour un nombre suffisant de valeurs, l'avantage du distingueur non-adaptatif $Adv_E^{\mathsf{PRP}}(\mathcal{A})$ sera non négligeable. Ce distingueur doit également fonctionner pour une très large partie de l'espace des clés.

Algorithm 1 Un distingueur générique non-adaptatif limité à n requêtes

Paramètres : une complexité n, une région d'approbation $A^{(n)}$
Oracle : Un oracle $\mathcal{O}$ implémentant une permutation c
  Pour un ensemble de messages $\mathbf{P} = (P_1, \cdots, P_n)$
  Demander $\mathbf{C} = (C_1, \cdots, C_n) = c(P_1, \cdots, P_n)$ à $\mathcal{O}$
  Si $\mathbf{C} \in A^{(n)}$ **Alors**
    Sortir 1
  Sinon
    Sortir 0
  Fin Si

Ce distingueur est générique et comprend les cas suivants : clairs connus (KPA), clairs choisis (CPA) et en modifiant légèrement la définition de l'oracle $\mathcal{O}$ peut aussi inclure les cas chiffrés connus (KCA) et chiffrés choisis (CCA). Notons cependant que dans le cas d'une attaque à clairs (ou chiffrés) choisis, il est nécessaire de filtrer l'ensemble des textes choisis en entrée à l'aide d'un filtre particulier prédéfini h_1. La notation $R(\mathbf{P}, \mathbf{C})$ devient donc $R(h_1(\mathbf{P}), \mathbf{C})$ dans le cas de clairs choisis.

Le but du cryptanalyste va donc être de chercher et de trouver une "bonne" relation R.

Distingueur différentiel

Dans un distingueur différentiel, le but de l'adversaire (comme expliqué dans [Vau03, Jun03]) est de trouver une relation R telle que : étant donné le filtre $h_1(\mathbf{P}) = ((P_1, P_1 + a), (P_2, P_2 + a), \cdots, (P_n, P_n + a))$ où a est une différence choisie et où $+$ est la loi additive sur l'ensemble d'entrée, il existe une paire de textes chiffrés correspondants $(E_K(P_i), E_K(P_i + a))$ telle que $E_K(P_i) + E_K(P_i + a) = b$ où b est une différence choisie et où $+$ est la loi additive sur l'ensemble de sortie. L'avantage du distingueur correspondant s'écrit (voir [Vau03]) :

$$Adv_E^{\mathsf{CPA}}(\mathcal{A}) \leq n \cdot \max(\frac{1}{2^l - 1}, EDP^E(a, b))$$

où $\frac{1}{2^l - 1}$ est la distribution uniforme sur $\{0,1\}^l$ de la permutation aléatoire G et où $EDP^E(a, b)$ est la moyenne prise sur l'ensemble des clés de la valeur DP^E :

$$EDP^E(a, b) = Esp_{K \in \mathcal{K}}(DP^{E_K}(a, b)) \text{ avec } DP^{E_K}(a, b) = Pr_P[E_K(P + a) = E_K(P) + b]$$

Ce distingueur différentiel est celui décrit dans [BS91] et ne tient pas compte des généralisations qui ont suivi telles que la différentielle tronquée [Knu95], la différentielle impossible [BBS99a] ou la différentielle d'ordre supérieur [Knu95].

Distingueur linéaire

Dans un distingueur linéaire (comme montré dans [Vau03]), le but de l'adversaire est de trouver une relation linéaire R telle que : étant donné $\mathbf{P} = (P_1, \cdots, P_n)$ et les chiffrés correspondants $\mathbf{C} = (C_1, \cdots, C_n)$, il existe un sous-ensemble $\mathcal{U}$ de u valeurs tel que $\forall i \in \mathcal{U}, a \cdot P_i = b \cdot C_i$ et $(u \neq n/2)$ où $\cdot$ désigne le produit scalaire usuel de $\{0,1\}^n$ dans $\{0,1\}$. La cryptanalyse linéaire a été introduite par H. Gilbert et A. Tardy-Corfdir dans [TCG91] contre l'algorithme FEAL puis appliquée par M. Matsui [Mat93, Mat94] au DES. L'avantage du distingueur correspondant s'écrit (voir [Vau03]) :

$$\lim_{n\to\infty} \frac{Adv_E^{\mathsf{KPA}}(\mathcal{A})}{n^{1/3}} \leq 9.3 \left(\max(\frac{1}{2^l - 1}, ELP^C(a,b))\right)^{1/3}$$

où $\frac{1}{2^l-1}$ est la distribution uniforme sur $\{0,1\}^l$ de la permutation aléatoire G et où $ELP^C(a,b)$ est la moyenne prise sur l'ensemble des clés de la valeur LP^C :

$$ELP^C(a,b) = Esp_{K\in\mathcal{K}}(LP^{E_K}(a,b)) \text{ avec } LP^{E_K}(a,b) = (2 \cdot Pr(a \cdot P = b \cdot C_K(P)) - 1)^2$$

Distingueur intégral

Dans un distingueur intégral [KW02, BS01], le but de l'adversaire est de trouver une relation R telle que : étant donné le filtre $h_1(\mathbf{P}) = \int S = (P_1, P_2, \cdots P_n)$ tel que tous les textes clairs appartiennent à un sous-ensemble particulier S, l'ensemble correspondant de textes chiffrés vérifie $\int_{\mathbf{P}\in S} E_K(\mathbf{P}) = c$ où c est une valeur connue et où le symbole $\int$ désigne une loi particulière (habituellement l'addition ou la multiplication) sur la structure algébrique sous-jacente utilisée par le chiffrement. Dans ce type de distingueur, la probabilité d'occurrence d'une telle propriété est égale à 1 et l'existence de propriétés intégrales pour un chiffrement dépend beaucoup de sa structure.

L'attaque la plus célèbre utilisant une propriété intégrale reste celle dans le modèle UK contre 6 tours de l'AES proposée dans la soumission originale [DR98] utilisant comme structure de base le corps fini sous-jacent $GF(2^8)$ et étendue à 8 tours dans [FKL+01] utilisant une propriété intégrale dite d'ordre $d = 4$, c'est-à-dire, si un mot peut prendre m valeurs dans une propriété intégrale d'ordre 1 alors la propriété intégrale d'ordre d sera celle où un mot particulier prend toutes les valeurs exactement m^{d-1} fois. On peut également citer les attaques à clés inconnues contre SQUARE [DKR97] ou SAFER [BCD03] et bien sûr l'attaque intégrale dans le modèle à clés connues présentée dans [KR07].

Le lien existant entre différentielles tronquées, différentielles d'ordre supérieur et propriétés intégrales n'est pas bien fixé et dépend du chiffrement : dans certains cas, les propriétés intégrales contiennent à la fois des différentielles tronquées et des différentielles d'ordre supérieur mais fonctionnent souvent moins bien lorsque les composants du chiffrement ne sont pas bijectifs.

Afin s'illustrer ce qu'est un distingueur utilisant une propriété intégrale, nous prendrons ici le fameux exemple du chiffrement E_K représentant trois tours de l'AES [DR98]. Soit $h_1(\mathbf{P}) = \{P_i\}_{i=0}^{255}$ un ensemble de 256 textes clairs prenant toutes les valeurs possibles entre 0 et 255 sur un octet particulier (dit octet actif) et constant sur tous les autres octets. L'ensemble $\mathbf{C} = \{C_i\}_{i=0}^{255}$ est l'ensemble de textes chiffrés correspondants i.e. $\mathbf{C} = E_K(\mathbf{P})$, où les C_i représentent la concaténation des 16 octets de sortie : $C_i = C_{i,0}||C_{i,1}||\cdots||C_{i,15}$. Alors, la relation $R(h_1(\mathbf{P}), \mathbf{C})$ est $\bigoplus_{i=0}^{255} C_{i,j}$ pour $j = 0 \cdots 15$ avec $\mathbf{P} = \{P_i\}_{i=0}^{255}$ et $\mathbf{C} = E_K(h_1(\mathbf{P}))$ telle que si $\bigoplus_{i=0}^{255} C_{i,j} = 0$ pour $j = 0 \cdots 15$, le distingueur renvoie 1 ; sinon, il renvoie 0. Si E_K est implémenté dans le distingueur, la probabilité que le distingueur renvoie 1 en testant $R(h_1(\mathbf{P}), \mathbf{C})$ est 1, tandis que

pour une permutation aléatoire G, cette probabilité est 2^{-l}. Ainsi, $\mathbf{Adv}^{\mathsf{PRP\text{-}CPA}}_{E_K,G}(\mathcal{A}) = 1-2^{-l} >> \varepsilon$.

Nouvelles formes d'attaques

Alors que les stratégies conventionnelles de la cryptanalyse telles que les distingueurs différentiel ou linéaire présentés précédemment considèrent l'analyse d'une version réduite $\tilde{E}$ d'un chiffrement E en un seul morceau, d'autres stratégies d'attaques ultérieures considère que le chiffrement $\tilde{E}$ peut être modélisé en deux parties : $\tilde{E} = E_1 \circ E_0$. C'est le cas par exemple des attaques différentielle-linéaire [LH94], miss-in-the-middle [BBS99b], boomerang (amplifié) [Wag99, KKS01] ou des techniques inside-out [Wag99].

Plus récemment, des techniques comme les distingueurs par rebond [MRST09] ou les distingueurs "biclique" [BKR11] considèrent un chiffrement réduit découpé en trois parties : $\tilde{E} = E_c \circ E_b \circ E_a$. Cela signifie que chaque partie E_i ($i \in \{a, b, c\}$) est plus courte que les E_j ($j \in \{0, 1\}$) précédents et que le nombre de points permettant de découper un chiffrement en 3 au lieu de 2 parties est clairement plus grand.

2.2.3 Distingueurs dans le modèle KK

Dans [MPP09] (cet article est également présenté en Annexe A), avec R. Phan et B. Pousse, nous avons tenté de formaliser la notion de distingueur pour le modèle KK. Nous l'avons décrit comme un distingueur non-adaptatif à textes du milieu choisis. Ce distingueur est décrit à l'algorithme 2. Dans ce cas, l'oracle $\mathcal{O}$ implémente soit E_K soit G avec une probabilité 1/2. La région d'approbation $A^{(n)}$ dépend ici de l'ensemble des valeurs d'entrée $\mathbf{P} = (P_1, \cdots, P_n)$ et de l'ensemble des valeurs de sortie correspondantes $\mathbf{C} = (C_1, \cdots, C_n)$ calculées à partir d'un ensemble de textes du milieu particulier $\mathbf{M} = (M_1, \cdots, M_n)$. Si une relation particulière $R(\mathbf{P}, \mathbf{C})$ qui définit la région d'approbation $A^{(n)}$ existe pour un nombre suffisant de valeurs, l'avantage du distingueur non-adaptatif $Adv^{\mathsf{PRP\text{-}CMA}}_E(\mathcal{A})$ sera alors non-négligeable. Comme les distingueurs précédents, ce distingueur doit fonctionner pour une très grande partie des clés. Dans le modèle à clés connues, les fonctions f_1 et f_2 sont publiques car elles ne peuvent en aucun cas être dépendantes de la clé sinon la relation $R(\cdot, \cdot)$ devient impossible à calculer et donc à vérifier.

Algorithm 2 Un distingueur générique non-adaptatif à textes du milieu choisi limité à n requêtes (PRP-CMA)

Paramètres : une complexité n, une région d'approbation $A^{(n)}$
Oracle : un oracle $\mathcal{O}$ implémentant deux fonctions internes f_1 (resp. f_2) de la permutation c qui prennent en entrée un texte du milieu et calculent le texte clair (resp. chiffré) correspondant
  Choisir un ensemble de textes du milieu $\mathbf{M} = (M_1, \cdots, M_n)$
  Demander $\mathbf{P} = (P_1, \cdots, P_n) = (f_1(M_1), \cdots, f_1(M_n))$ et $\mathbf{C} = (C_1, \cdots, C_n) = (f_2(M_1), \cdots, f_2(M_n))$ à $\mathcal{O}$
  Si $(\mathbf{P}, \mathbf{C}) \in A^{(n)}$ **Alors**
    Sortir 1
  Sinon
    Sortir 0
  Fin Si

Cette notion permet de capturer celle définie dans [KR07] car s'il existe un distingueur d'avantage non-négligeable dans le modèle précédent alors il fournit une propriété particulière liant les

couples d'entrées/sorties du chiffrement ayant une probabilité de se produire significativement différente de la probabilité uniforme. Cette notion permet aussi de formaliser les distingueurs utilisés dans [MPRS09, GP10] où sont présentées des attaques par rebond contre des chiffrements par blocs dans le modèle KK et contre des fonctions de compression utilisées dans des soumissions de la compétition SHA-3 où, dans le mode Davies-Meyer, la clé (i.e. le message) est connue. Pour plus de détails concernant à la fois une discussion plus approfondie sur cette notion et le lien avec la "correlation intractability" définie par Canetti et al. dans [CGH04], nous renvoyons le lecteur à l'Annexe A ou à [MPP09]. Notons également que les distingueurs à clés connues identifiés à ce jour peuvent être convertis en un distingueur de cette forme.

Notons également que, comme remarqué dans [SY11], le distingueur développé ici n'est pas suffisamment précis pour pouvoir transformer une attaque à clés connues contre un chiffrement par blocs en une attaque autre qu'une attaque par distingueur sur une fonction de compression, c'est-à-dire qu'il est impossible en l'état actuel de déduire de ce modèle une attaque par collisions ou par preimage.

2.2.4 Distingueurs dans le modèle RKK

Dans [MP], nous avons également défini un nouveau modèle d'adversaire pour un type particulier de distingueur que nous appelons distingueur RKK. Le principe général de ce distingueur est de combiner des clés liées avec des textes du milieu choisis afin de construire des propriétés permettant de monter des attaques contre des chiffrements par blocs ou des fonctions de compression quand la clé est le message. Plus précisément, il s'agit ici, étant donné une clé connue donnée K, de calculer un sous-ensemble particulier de clés $\mathbf{K}$ dépendant de la clé K et de construire à partir de textes du milieu dépendant du sous-ensemble choisi $\mathbf{K}$ les ensembles d'entrées/sorties correspondants reliés ensemble par une relation R. Celle-ci permet de distinguer le sous-ensemble créé d'une permutation aléatoire. Plus précisément, on distingue une famille particulière de permutations construites d'une famille de permutations aléatoires. Ce distingueur est décrit à l'algorithme 3. L'oracle utilise les clés et les textes du milieu fournis par l'adversaire dans une ou deux directions afin de produire les textes clairs et/ou chiffrés correspondants.

Algorithm 3 Un distingueur générique non-adaptatif à clés reliées connues et à textes du milieu choisis limité à n requêtes (PRP-RKCMA)

Paramètres : une complexité n, une région d'approbation $A^{(n)}$
Oracle : un oracle $\mathcal{O}$ implémentant deux fonctions internes f_1 (resp. f_2) de la permutation c qui prennent en entrée des textes du milieu et calcule les textes clairs (resp. chiffrés) correspondants

A partir de la clé connue K, construire des clés particulières $\mathbf{K} = (K_1, \cdots, K_m)$ et des textes du milieu particuliers $\mathbf{M} = (M_1, \cdots, M_n)$
Demander $\mathbf{P} = (P_1, \cdots, P_{nm}) = (f_1(K_1, M_1), \cdots, f_1(K_1, M_n), \cdots, f_1(K_m, M_1), \cdots, f_1(K_m, M_n))$ et $\mathbf{C} = (C_1, \cdots, C_{nm}) = (f_2(K_1, M_1), \cdots, f_2(K_1, M_n), \cdots, f_2(K_m, M_1, \cdots, f_2(K_m, M_n))$ à $\mathcal{O}$
Si $(\mathbf{P}, \mathbf{C}) \in A^{(n)}$ **Alors**
  Sortir 1
Sinon
  Sortir 0
Fin Si

Comme dans le cas du distingueur à clés connues, les fonctions f_1 et f_2 sont publiques et indépendantes de la clé. Cette notion est très proche de celle utilisée pour le distingueur

correspondant à la première attaque de [BKN09] où les auteurs choisissent un ensemble particulier de clés reliées et des textes clairs choisis pour leurs bonnes propriétés différentielles afin d'obtenir des sommes nulles sur les textes clairs et les textes chiffrés ainsi obtenus. Cependant, cette première attaque ne fonctionne que sur un ensemble réduit de clés donc des clés faibles, tandis que la deuxième attaque est une attaque à clés reliées classique comme celles décrites dans [BK09].

2.2.5 Distingueurs dans le modèle CK

Dans [MP], nous avons également tenté de définir le distingueur utilisé dans la deuxième attaque présentée dans [BKN09]. Celui-ci est présenté à l'algorithme 4. Le principe général est de combiner dans ce cas des clés reliées choisies et des textes clairs ou du milieu choisis afin de construire des propriétés particulières permettant de monter des attaques contre des algorithmes en considérant qu'ils sont utilisés comme des fonctions de hachage. Dans [BKN09], les auteurs choisissent des clés particulières et des textes clairs particuliers avec de bonnes propriétés différentielles afin d'obtenir des sommes nulles sur les textes clairs et les textes chiffrés. Cette attaque conduit à un distingueur complet sur les 14 tours de l'AES-256.

Dans l'algorithme 4, nous définissons le distingueur utilisé comme étant un distingueur non-adaptatif à clés choisies et à textes du milieu choisis. L'oracle utilise les clés et les textes du milieu choisis par l'adversaire dans une ou deux directions afin de produire les textes clairs et/ou chiffrés correspondants.

Algorithm 4 Un distingueur générique non-adaptatif à clés reliées choisies et à textes du milieu choisis limité à n requêtes (PRP-CKCMA)

Paramètres : une complexité n, une région d'approbation $A^{(n)}$
Oracle : un oracle $\mathcal{O}$ implémentant deux fonctions internes f_1 (resp. f_2) de la permutation c qui prennent en entrée des textes du milieu et calcule les textes clairs (resp. chiffrés) correspondants

Calculer des clés particulières $\mathbf{K} = (K_1, \cdots, K_m)$ et des textes du milieu particuliers $\mathbf{M} = (M_1, \cdots, M_n)$
Demander $\mathbf{P} = (P_1, \cdots, P_{nm}) = (f_1(K_1, M_1), \cdots, f_1(K_1, M_n), \cdots, f_1(K_m, M_1, \cdots, f_1(K_m, M_n))$ et $\mathbf{C} = (C_1, \cdots, C_{nm}) = (f_2(K_1, M_1), \cdots, f_2(K_1, M_n), \cdots, f_2(K_m, M_1, \cdots, f_2(K_m, M_n))$ à $\mathcal{O}$

Si $(\mathbf{P}, \mathbf{C}) \in A^{(n)}$ **Alors**
  Sortir 1
Sinon
  Sortir 0
Fin Si

Dans l'attaque proposée dans [BKN09], la fonction f_1 est en fait l'identité ce qui est un cas particulier du distingueur présenté ici.

2.2.6 Exemples d'application

Rijndael

La famille de chiffrement par blocs Rijndael-b peut être considérée comme celle constituée des grands frères de l'AES. En effet, dans la soumission initiale au NIST [DR98], la famille de chiffrements proposée était constituée de Rijndael-128 devenu l'AES [FIP01] et de 4 autres

versions Rijndael-160, Rijndael-192, Rijndael-224 et Rijndael-256 [DR02] qui prenaient en entrée des blocs de tailles respectives 160, 192, 224 et 256 bits, la fonction de tour elle-même et le cadencement de clé étant très peu modifiés, seul change le nombre d'étages.

Dans un premier temps, nous avons donc cherché des propriétés intégrales sur 4 ou 5 étages de chacune des versions de Rijndael ayant de meilleures complexités que celle sur 4 tours de l'AES nécessitant 2^{32} clairs avec 4 octets actifs et 2^{32} calculs afin de produire les chiffrés correspondants dont la somme pour chaque octet prise sur les 2^{32} valeurs d'entrée est nulle. Parmi toutes celles que nous avons obtenues, on peut citer ici celle sur 5 tours de Rijndael-256 qui utilise 2^{32} clairs et 2^{32} calculs et qui aboutit à des sommes nulles en sortie sur 8 octets. Cette première propriété nous a permis de monter une attaque à clés inconnues décrite dans [GM08] contre 9 tours de Rijndael-256 sous une clé de 192 bits et nécessitant $2^{128} - 2^{119}$ textes clairs choisis pour une complexité de 2^{188} chiffrements. Les autres attaques à clés inconnues contre les différentes variantes de Rijndael sont décrites dans [MP09].

Dans un deuxième temps afin de pouvoir monter des attaques à clés connues contre les différentes versions de Rijndael, nous avons cherché des propriétés intégrales sur le chiffrement inverse. La propriété de base concernant l'AES et utilisée dans [KR07] est constituée de 3 étages en sens inverse et nécessite 2^{32} clairs avec 4 octets actifs et 2^{32} calculs afin de produire les déchiffrés correspondants dont la somme pour chaque octet prise sur les 2^{32} valeurs d'entrée est nulle. Il est à noter que c'est en combinant la propriété sur 4 tours de l'AES en sens direct et celle sur 3 tours de l'AES en sens indirect que L. Knudsen et V. Rijmen dans [KR07] obtiennent un distingueur à clés connues sur 7 tours de l'AES nécessitant 2^{56} textes du milieu et 2^{56} calculs. La relation R liant les textes clairs et les textes chiffrés est telle que la somme pour chaque octet prise sur les 2^{56} valeurs de textes clairs (générées à partir des textes du milieu choisis) et la somme pour chaque octet prise sur les 2^{56} valeurs de textes chiffrés correspondants sont nulles. La complexité pour obtenir une telle structure pour une permutation aléatoire peut être approximée par le "k-sum problem" décrit dans [Wag02] et est de 2^{58} calculs.

Nous avons trouvé que pour Rijndael-256, il est possible d'obtenir une propriété intégrale sur 3 tours du chiffrement inverse nécessitant 2^{16} textes du milieu et 2^{16} calculs aboutissant à des sommes nulles en entrée sur 28 octets. il est ainsi possible de monter un distingueur à clés connues contre 8 tours de Rijndael-256 en utilisant 2^{40} textes du milieu choisis avec une complexité de 2^{40} chiffrements, la relation R à vérifier étant la même que précédemment : la somme sur chacun des 24 octets particuliers prise sur les 2^{40} valeurs de textes clairs (générées à partir des textes du milieu choisis) et la somme sur chacun des 8 octets particuliers prise sur les 2^{40} valeurs de textes chiffrés correspondants sont nulles. Nous obtenons des distingueurs du même type avec des complexités supérieures pour les autres versions de Rijndael. Ces résultats ont été présentés à la conférence Africacrypt 2009 [MPP09] et sont donnés en Annexe A.

En conclusion de cette étude, il semble que tant que le nombre d'applications de la partie linéaire d'un chiffrement par blocs orienté mots n'a pas atteint sa borne MDS (Maximum Distance Separable) r, il est possible de construire des propriétés intégrales d'ordre d, $d \geq 1$ sur au moins $r+1$ étages. Ce que j'appelle ici borne MDS est défini comme le nombre i nécessaire d'applications successives de la partie linéaire d'un chiffrement par blocs avant que le code correcteur $(X, L^i(X))$ ne devienne MDS. Même si je n'arrive pas à démontrer ce résultat, il semble se réaliser dans la pratique. Par exemple, il faut 3 applications de la partie linéaire de l'AES pour en faire un code MDS et l'AES possède une propriété intégrale sur 4 tours d'ordre 4 tandis que Rijndael-256 atteint sa borne MDS en 4 tours et possède une propriété intégrale d'ordre 16 sur 6 tours. Cette conjecture permet donc d'obtenir une borne minimale à partir de laquelle chercher une propriété intégrale. Notons également que le nombre de tours que semble atteindre une attaque par rebond comme celles proposées dans [Sas10] ou [GP10] dans le modèle KK restera toujours plus grand

que dans le cas d'une attaque intégrale.

L'AES

Dans [MP], nous présentons deux attaques contre l'AES-128, il s'agit de deux distingueurs, le premier dans le modèle RKK et le deuxième dans le modèle CK. L'idée principale de ces distingueurs est d'utiliser le cadencement de clé de l'AES afin de "gagner" un tour en remontant car le cadencement de clé de l'AES est complètement bijectif. Il s'agit donc de choisir un ensemble de clés liées qui vérifie la propriété intégrale en sens inverse sur deux tours pour la sous-clé K_2 et d'annuler les effets de la sous-clé K_3 à l'aide des textes du milieu choisis. Cela permet de construire un distingueur dans le modèle RKK vrai pour 2^{122} clés sur 7 étages de l'AES-128 utilisant 2^{40} textes du milieu sous 2^8 clés liées pour une complexité de 2^{40} chiffrements qui teste si la somme sur chacun de 8 octets particuliers d'entrée est nulle et si la somme sur chacun des octets de sortie est également nulle. Ce distingueur a donc une meilleure complexité que celui proposé dans [KR07]. Il est également possible de monter dans le modèle CK un distingueur sur 8 étages de l'AES-128 qui fonctionne pour 2^{105} clés. Ici, l'ensemble de sous-clés K_3 liées considéré comportera une pleine colonne d'octets actifs tandis que les textes du milieu choisis permettront d'annuler les effets des sous-clés déduites K_4. On obtient ainsi dans le modèle CK un distingueur sur 8 étages de l'AES-128 utilisant 2^{64} textes du milieu sous 2^{32} clés ayant une complexité de 2^{64} chiffrements qui teste si la somme sur chacun de 4 octets particuliers d'entrée est nulle et si la somme sur chacun des octets de sortie est également nulle. Ce résultat est moins bon que le distingueur présenté dans [GP10], mais reste cependant intéressant car il n'utilise pas de mémoire.

Il est à noter que les résultats de cette soumission semblent de moins en moins pertinents au regard des dernières attaques publiées contre l'AES à Asiacrypt 2011 [BKR11]. Nous espérons donc que cet article soit rapidement accepté.

2.3 Cryptanalyse et design de fonctions de hachage

Mon intérêt pour les fonctions de hachage est très récent et date du début de la compétition SHA-3, c'est pourquoi cette section pourra paraître légèrement dépeuplée par rapport aux autres.

2.3.1 Distingueurs et Collisions sur des fonctions de hachage

Distingueurs intégraux contre Hamsi-256, LANE-256 et Grøstl-512 v1

Avec B. Pousse et R. Phan, nous avons étudié les propriétés intégrales de 3 candidats particuliers de la compétition SHA-3 : il s'agit de Hamsi-256 [Küc09] finaliste du deuxième tour puis écarté, LANE-256 [Ind08] écarté dès le deuxième tour et Grøstl-512 v1 [GKM+08] finaliste du troisième tour. Ces résultats ont été publiés à la conférence CANS 2010 [MPP10].

Il s'agissait tout d'abord de rechercher parmi les soumissions à SHA-3 lesquelles maximisaient la borne MDS. C'est ainsi que ce sont dégagées ces trois fonctions. Puis, pour chacune de ces trois fonctions de compression, nous avons appliqué la méthode décrite au paragraphe 2.2.6 : en partant au milieu de la fonction de compression, nous avons cherché des propriétés intégrales dans le sens direct et également dans le sens inverse pour ensuite combiner ces propriétés dans les deux sens afin de monter des distingueurs intégraux. Je ne donnerai ici que les distingueurs sans donner le détail de chacune des propriétés utilisées. Le lecteur peut se reporter à [MPP10].

Dans le cas d'Hamsi-256, nous avons trouvé un distingueur intégral sur la transformation finale P_f qui utilise 2^{16} textes du milieu constants partout sauf sur les 4 bits de poids faible de

4 mots particuliers, qui teste si la somme prise sur l'ensemble de ces valeurs pour chacun des mots de 32 bits en entrée et en sortie est nulle et qui nécessite 2^{16} appels à P_f. La complexité de ce distingueur est meilleure que celui proposé dans [AKK+10] qui nécessite 2^{28} textes du milieu et 2^{28} appels à P_f. Notons cependant que la meilleure attaque à ce jour contre Hamsi-256 est celle présentée par T. Fuhr à Asiacrypt 2010 [Fuh10] qui est capable de trouver des secondes preimages sur des condensés de messages avec une complexité de $2^{251.3}$ calculs grâce à l'utilisation d'équations affines sur les entrées/sorties de la fonction de compression.

Pour LANE-256, nous avons monté un distingueur intégral qui utilise 2^{112} textes du milieu répété pour 2^{128} valeurs de chaînage où les permutations P_0, P_1 et P_2 comportent 3 tours à la place des 6 requis dans la version complète de LANE-256. Ce distingueur intégral teste si la somme prise sur l'ensemble de ces valeurs pour chacun des octets en entrée et en sortie est nulle. Il a une complexité de 2^{240} appels à la fonction de compression. Le même type de distingueur existe également contre LANE-512. Ce distingueur est bien sûr moins bon que le meilleur résultat de cryptanalyse contre LANE-256 [MNPN+09] qui présente des collisions semi-libres contre la version complète de la fonction de compression LANE-256 nécessitant 2^{96} calculs et utilisant 2^{88} bits en mémoire. Des collisions semi-libres peuvent également être trouvées contre la version complète de la fonction de compression avec 2^{224} appels à la fonction de compression et une mémoire de 2^{128} bits. Ces deux attaques utilisent des techniques d'attaques par rebond (qui seront décrites au paragraphe suivant).

Dans le cas de Grøstl-512 v1, nous avons tout d'abord monté un distingueur intégral sur chacune des permutations P et Q limitées à 10 étages au lieu des 14 requis. Ce distingueur utilise 2^{512} textes du milieu avec une complexité de 2^{512} calculs et teste si la somme prise octet par octet sur 24 octets particuliers est nulle. Ces deux distingueurs peuvent être combinés afin de monter un distingueur intégral contre la fonction de compression elle-même lorsque P et Q possèdent 10 tours. La complexité d'un tel distingueur est de 2^{513} opérations avec peu de mémoire requise afin de trouver des sommes nulles sur des octets particuliers (7 octets en sortie et 24 octets en entrée). Ce nouveau distingueur améliore celui décrit dans la proposition originale [GKM+08] qui atteignait 9 tours en 2^{704} opérations. Cependant, la meilleure attaque contre Grøstl-512 reste celle de T. Peyrin présentée à Crypto 2010 qui permet de construire un distingueur utilisant des différentielles tronquées contre une version à 11 tours de Grøstl-512 nécessitant 2^{640} calculs et une mémoire de 2^{64} bits (ici, c'est également une attaque par rebond qui est utilisée). C'est d'ailleurs cet article qui a forcé les auteurs de Grøstl v1 à modifier leur soumission au NIST, Grøstl v2 [GKM+11], pour la dernière phase de la compétition SHA-3.

Collisions et distingueurs contre des versions réduites de SHAvite-3-256

Avec T. Peyrin et M. Naya-Plasencia, nous avons dans [MNPP11] présenté des attaques contre des versions réduites de la fonction SHAvite-3-256 [BD09] en utilisant les cryptanalyses par rebond et par Super-boîte S. Nous avons ainsi pu construire des distingueurs par sels liés choisis sur la fonction de compression de SHAvite-3-256 jusqu'à 8 étages (elle en comporte 12 dans sa version complète) et des collisions libres jusqu'à sept tours.

Je ne détaillerai pas ici la fonction SHAvite-3-256, finaliste du deuxième tour de la compétition SHA-3 et éliminée au troisième, le lecteur peut trouver sa description complète dans [BD09] ou dans [MNPP11]. Précisons seulement que SHAvite-3-256 utilise une construction HAIFA [BD07] : $h_0 = IV, h_i = C_{256}(h_{i-1}, M_{i-1}, salt, cnt)$ et $hash = trunc_n(h_i)$ où la fonction de compression C_{256}, utilisée en mode Davies-Meyer, prend en entrée la valeur de chaînage de 256 bits h_{i-1}, un bloc de message de 512 bits M_{i-1}, un sel aléatoire *salt* de 256 bits et un compteur de 64 bits *cnt*.

On peut juste signaler que la fonction de compression C_{256} est composée d'un chiffrement par blocs E^{256} utilisant un schéma de Feistel sur 12 étages où 3 tours de l'AES font office de fonction étage et d'un cadencement de clé permettant de générer à partir du bloc de message, du sel et du compteur toutes les sous-clés nécessaires. Le cadencement de clé utilise 4 tours de l'AES en parallèle composés avec deux étages de diffusion linéaire.

Dans la suite de ce paragraphe, nous donnerons quelques détails sur les distingueurs construits. Pour cela, nous introduisons auparavant les principes des attaques rebonds et Super-Boîte S.

Principes des attaques par rebond

L'attaque par rebond [MRST09] utilise des différentielles tronquées : au lieu de regarder la valeur exacte d'une différence sur un octet particulier, on se contente de savoir si un octet contient (octet actif) ou non (octet inactif) une différence. Dans ce cas, le passage dans les boîtes S des différences devient déterministe tandis que cette transition différentielle dans la partie linéaire devient probabiliste. Si on considère le cas de la fonction $MixColumns$ de l'AES, les probabilités induites dépendent directement de sa propriété MDS (le nombre d'octets actifs en entrée/sortie d'une colonne sera toujours supérieur ou égal à 5). Ainsi, en tirant aléatoirement des valeurs d'entrées, la probabilité de succès d'une transition différentielle à travers $MixColumns$, sous la contrainte MDS, est déterminée par le nombre d'octets actifs en sortie : si une transition différentielle contient k octets actifs en sortie sur une colonne, sa probabilité de succès sera égale à $2^{-8\times(4-k)}$. Par exemple, une transition $4 \mapsto 1$ d'une colonne a une probabilité de succès égale à 2^{-24} (elle est la même pour $MixColumns^{-1}$). Cette transition reste la plus usitée quand un chiffrement utilise une opération $MixColumns$. Ainsi, sur 3 tours de l'AES, la transition différentielle la plus classique est celle présentée à la Fig. 2.1 qui a une probabilité de se produire égale à 2^{-24} quand les valeurs d'entrée et les valeurs de différences sont prises aléatoirement.

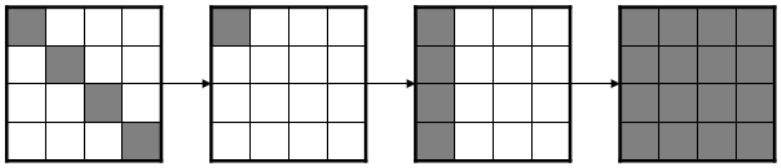


FIGURE 2.1 – effet d'un chemin de différentielles tronquées sur 3 tours de l'AES.

Afin de monter une attaque par rebond sur un nombre de tours plus conséquent, il est nécessaire de prendre en compte les degrés de liberté [MRST09] restant dans la fonction. L'attaque par rebond utilise une technique de rencontre dans le milieu pour faire converger les chemins différentiels à l'endroit où cette rencontre sera la moins coûteuse. Cette technique permet d'attaquer 2 tours de l'AES dans le milieu. Dans [GP10], H. Gilbert et T. Peyrin ont réussi à gagner un troisième tour dans le milieu via une technique dite de Super-boîte S, l'idée étant de considérer deux tours de l'AES comme l'application parallèle d'un étage de grosses boîtes S (les Super-boîtes S) précédées et suivies de transformations affines. Cette méthode permet de trouver plusieurs solutions de différentielles tronquées valides sur trois tours de l'AES pour un coût moyen de 1. Cependant, il est nécessaire de payer un coût minimal : dans le cas de l'AES, la complexité de l'attaque est de $\max\{2^{32}, k\}$ calculs et 2^{32} en mémoire, où k est le nombre de solutions trouvées vérifiant la partie du milieu du chiffrement.

Distingueurs par Super-boîte S contre 7 tours de SHAvite-3-256

Comme la fonction étage utilisée dans SHAvite-3-256 est composée de 3 tours de l'AES, il

semblait naturel d'appliquer la méthode de la Super-boîte S à ce cas précis. Si on considère la fonction E^{256}, il est possible de construire un distingueur sur sept étages de la manière suivante : on fixe tout d'abord une différence Δ sur les 4 octets de la diagonale en entrée de la partie droite du troisième tour tandis que sa partie gauche ne contient pas de différence du tout. On applique la technique de Super-boîte S sur les tours 3 et 4 afin de trouver 2^{32} solutions valides pour passer chacun de ces tours. Ainsi, on obtient 2 ensembles de 2^{32} valeurs qui vérifient indépendamment le troisième et le quatrième étages. Le reste du chemin différentiel est construit de façon probabiliste : la probabilité de remonter avec un chemin différentiel intéressant le deuxième et le cinquième tours est égale à 1 tandis que la probabilité de remonter le premier et le sixième étage est égale à $(2^{-24})^2 = 2^{-48}$ afin d'obtenir une entrée où la partie droite est égale à Δ et la partie gauche est l'image par un tour d'AES d'une pleine colonne d'octets actifs, la sortie au septième étage étant quant à elle composée d'une partie gauche ayant la même forme et d'une partie droite non contrôlée. On obtient ainsi une solution qui vérifie le chemin différentiel complet en 2^{48} calculs et qui nécessite une mémoire de 2^{32}. La complexité pour trouver une telle structure pour une permutation aléatoire est de 2^{64} calculs (pour le détail du calcul dans le cas idéal, voir [GP10]).

Distingueurs à sels liés choisis contre 7 et 8 tours de SHAvite-3

Le principe de ces deux distingueurs est d'annuler à l'aide de l'algorithme de génération de sous-clés les différences sur les sous-clés des deuxième, troisième et quatrième étages en choisissant deux sels liés dépendants des différences introduites dans les blocs de messages à hacher. Ainsi le distingueur sur la fonction de compression se limitera à trouver une paire de valeurs et de différences vérifiant le chemin différentiel sur les tours 5, 6 et 7 en tenant compte des contraintes induites par l'algorithme de génération de sous-clés. Je ne détaillerai pas ici les tests à effectuer pour vérifier le chemin différentiel permettant de construire un distingueur sur sept étages de la fonction de compression dont la différence en entrée ne contient pas de différences sur sa partie droite et dont la différence en sortie ne contient pas de différences sur sa partie gauche, tandis que la différence sur le bloc de message est égale sur le premier et le troisième blocs de 128 bits, le lecteur peut se reporter à [MNPP11]. La complexité de ce distingueur est de 2^7 calculs.

Ce distingueur peut être également étendu à 8 étages en ajoutant un tour à la fin car il reste suffisamment de degrés de liberté dans l'algorithme de génération de sous-clés pour ajouter des contraintes. La complexité de ce distingueur est de 2^{25} calculs pour une mémoire de 2^{14}. Dans les deux cas, la construction de telles structures dans le cas idéal nécessiterait 2^{64} calculs.

Conclusion

On peut bien évidemment à partir de ces distingueurs sur 7 et 8 étages construire des collisions particulières sur 6 et 7 étages. La meilleure collision est une collision semi-libre à sels liés choisis sur 7 étages nécessitant 2^{96} calculs pour une mémoire de 2^{32}. Notons également que la version sur 512 bits de SHAvite-3, SHAvite-3-512, a été mise en danger par une attaque par preimage à compteur et sel choisis contre la version complète de la fonction de compression (voir [GLM+10]).

2.3.2 Design d'une fonction de hachage Lightweight : GLUON

Avec Thierry Berger, Joffrey D'Hayer et Kevin Marquet, nous nous sommes même commis dans la construction d'une fonction de hachage dite lightweigth (que l'on peut traduire par ayant une implémentation hardware compact) [BM] (l'article complet présentant la famille GLUON est donné en annexe A). L'idée était de suivre les premières propositions qui avaient été faites dans ce sens, à savoir QUARK [AHMNP10] et `PHOTON` [GPP11] qui ont une implémentation suffisamment compacte pour être utilisée sur des tags RFID. Afin de construire cette nouvelle proposition, qui comme les deux précédentes, tire son nom de la classification des particules élémentaires, nous

nous sommes surtout inspirés de QUARK en utilisant une construction éponge et une fonction de compression qui utilise des word ring FCSR comme ceux présentés dans le Chapitre 1 à la Section 1.2.4. Même si le design proposé ici est un petit peu plus lourd en terme de nombre de portes nécessaire à une implémentation matérielle (2071 portes pour l'instance la plus compacte) par rapport à QUARK et `PHOTON`, il nous a semblé intéressant de continuer à proposer de telles instances pour s'assurer un nombre de primitives suffisant.

Construction éponge

Les constructions éponge ont été proposées par Bertoni et al. dans [BDPA08] comme une nouvelle manière de construire des fonctions de hachage à partir d'une fonction ou d'une permutation fixée. Comme défini dans la Fig. 2.2, une construction éponge a un *taux* r correspondant à la taille des blocs de message à "entrer" dans la fonction via une loi de groupe, une *capacité* cp et une longueur de sortie N. La largeur de l'état interne b est définie comme $r + cp$ qui doit être plus grand que N. L'avantage dans de telles constructions comme signalé dans [GPP11] est que le rapport entre la taille de l'état interne et le niveau de sécurité est meilleur que dans les modes de hachage classiquement utilisés même si le hachage de petits messages n'est pas très efficace en raison de la phase de squeezing (que l'on pourrait traduire en français par phase de pression).

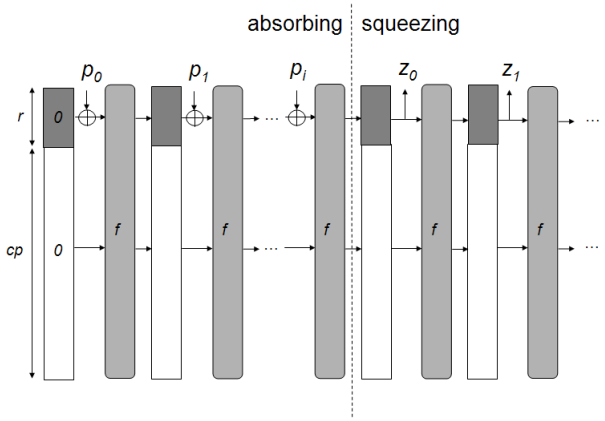


FIGURE 2.2 – Construction éponge.

Etant donné un état initial, la construction éponge traite un message M de longueur $|M| = \log_{2^r}(M)$ de la manière suivante :

1. **Initialisation :** Le message est complété par un '1' et suffisamment de 0 pour atteindre une taille multiple de r.
2. **Phase d'absorption :** Les blocs de message de r bits sont xorés avec r bits de l'état entre des applications de la fonction f.
3. **Phase de squeezing :** r bits de l'état sont sortis entre des applications de la fonction f jusqu'à ce que la sortie fasse au total N bits.

L'avantage d'une telle construction réside dans les preuves de sécurité fournies par les auteurs [BDPA08] dans le modèle d'indifférentiabilité de Maurer, Renner et Holenstein [MRH04]. Ainsi, il a par exemple été montré dans [BDPA08] que la probabilité de succès de différencier un oracle aléatoire d'une construction éponge faisant appel à une permutation ou une transformation aléatoire est bornée supérieurement par $1 - \exp(-Q^2 2^{-(c+1)})$ où Q est le nombre d'appels à f ou à son inverse si il existe. Pour cette borne particulière, les auteurs de [BDPA08] déduisent

une simplification du modèle de preuve dans le pire cas appelée *flat sponge claim* où des bornes concrètes de sécurité sont déduites en fonction de la capacité cp quant à la résistance aux collisions ou à la (deuxième) preimage.

Proposition pour la fonction de compression

Nous avons donc proposé la famille GLUON comme base d'une construction éponge où la fonction f utilise un word ring FCSR filtré linéairement. Cette fonction est directement inspirée des constructions F-FCSR-v3 et X-FCSR-v2 présentées au Chapitre 1. En deux mots, la structure générale de la famille GLUON est donnée à la Fig. 2.3 et est composée des éléments suivants :

- On note $m(t) = (m_0(t), \cdots, m_{w-1}(t))$ le contenu du registre principal du word ring FCSR de taille w mots de r bits et $c(t) = (c_0(t), \cdots, c_{w-1}(t))$ le registre de retenues avec a mémoires actives. (Ce FCSR est bien sûr défini par sa matrice de transition T).
- Un filtre xor-linéaire FI permettant de filtrer le contenu de $m(t)$ en xorant ensemble des versions décalées des mots $m_i(t)$ du registre principal recevant une rétroaction. Ce filtre est utilisé uniquement pour extraire une sortie de taille $(w-1) \times r$ bits.

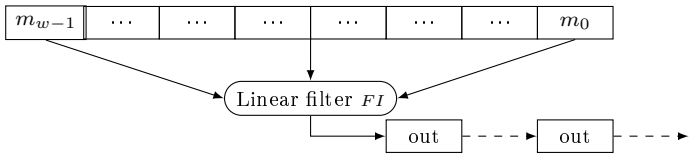


FIGURE 2.3 – Vue générale de la fonction f pour la famille GLUON.

La fonction f qui prend en entrée/sortie $b = (w-1) \times r$ bits fonctionne de la manière suivante :

1. *Initialisation :* Pour une entrée $s = (s_0, \cdots, s_{b-1})$, f initialise son état interne de la manière suivante :
   - Les $w-1$ premiers mots de m sont initialisés avec les $(w-1) \times r$ bits d'entrée.
   - Le dernier mot de m est initialisé avec la constante tout à 1 de longueur r.
   - Le registre de retenues est initialisé tout à 0.
2. *Mise à jour de l'état :* A partir de l'état interne $(m(t), c(t))$, le FCSR est clocké $d+4$ fois (où d est le diamètre du FCSR) à l'aide des équations : $m(t+1) = Tm(t) + c(t) \mod 2$ et $c(t+1) = Tm(t) + c(t) \div 2$.
3. *Calcul de la sortie :* Le FCSR est clocké $w-1$ fois. A chaque itération, un mot de r bits est extrait à l'aide du filtre linéaire FI afin d'obtenir $(w-1) \times r$ bits de sortie.

Le taux de l'éponge associé est r, sa capacité est $cp = (w-2) \times r$, la taille de f étant $b = r + cp$.

Nous laissons au lecteur le soin d'aller découvrir les différentes instances de GLUON ainsi que les arguments de sécurité pour la fonction f et les résultats de performances tant logiciel que matériel donnés en annexe A ou dans [BM].

2.4 Conclusion

Un sujet qui me tient particulièrement à cœur et qui n'a pas été présenté dans ce chapitre concerne l'apparition au cours de ces cinq dernières années d'un grand nombre de chiffrement par blocs dit ligthweight (au sens de "avec une implémentation matérielle compacte"). On peut

citer parmi toutes les propositions, PRESENT [BKL+07], HIGH [HSH+06], mCrypton [LK05] ou KATAN & KTANTAN [CDK09]. Leur étude tant du point de vue de leur sécurité que des possibles attaques à monter contre eux semble être au centre de l'actualité cryptographique. C'est un des buts du projet ANR INS BLOC accepté en juin 2011 et dont je suis coordinatrice. Nous souhaitons dans le cadre de ce projet être capable de définir de nouvelles preuves de sécurité fonction des nouveaux modèles de cryptanalyses existant contre les chiffrements par blocs. J'espère donc que la classification donnée dans ce chapitre permettra de simplifier cette tâche.

En ce qui concerne les fonctions de hachage, la communauté cryptographique continue d'étudier les finalistes de SHA-3 et attend avec anxiété la fin de cette compétition prévue en 2012.

Deuxième partie

Confiance dans les réseaux ambiants et solutions de sécurité pour les réseaux de capteurs sans fil

Chapitre 1

Confiance dans les réseaux ambiants

1.1 Introduction

A mon arrivée comme maître de conférence au laboratoire CITI en 2005, Stéphane Ubéda, alors directeur du laboratoire, m'a proposé de prendre la responsabilité du projet ANR ACI sécurité (2004-2007) KAA [9] (Knowledge Authentification Ambient) et de l'aider à encadrer la thèse de S. Galice [Gal07] financée par ce même projet. Il s'agissait d'un projet pluri-disciplinaire impliquant plusieurs laboratoires de différents domaines : le CITI bien sûr, l'institut des systèmes complexes (IXXI) qui est multi-établissements, le laboratoire d'économie des transports de l'ISH de Lyon, le CERDRID, laboratoire de droit de Saint-Étienne, l'ICTT (Interaction Collaborative - Téléformation - Téléactivités) de l'INSA de Lyon et le MAPPLY, laboratoire de mathématiques de l'INSA de Lyon.

Le but de ce projet était de définir à partir des interactions humaines de gestion de la confiance un modèle de confiance dans les réseaux ambiants qui soit informatiquement raisonnable et humainement acceptable. Les acteurs du projet issus des sciences humaines ont donc travaillé sur la définition de la confiance dans leur domaine. Ainsi, la sociologie s'est attachée à définir le fonctionnement de la confiance dans les groupes sociaux [Neu96], les juristes ont développé un point de vue alliant régulation juridique et confiance autour des modèles d'organisation juridique, les économistes ont travaillé sur la place de la confiance dans le contrat ainsi que sur le lien confiance/réciprocité observé dans les jeux de l'économie expérimentale [Dub06b]. En tant qu'informaticiens, nous avons tenté d'utiliser ces différentes visions pour construire un modèle de confiance pour les réseaux spontanés.

Les réseaux spontanés peuvent être vus comme une sous-catégorie des réseaux ambiants. Ces derniers sont des réseaux sans fil dans lesquels chaque entité peut se connecter partout, tout le temps si nécessaire, par l'intermédiaire d'objets communicants classiques dit intelligents comme un ordinateur portable ou un smartphone. Dans ce cas, seul l'utilisation d'un système distribué et non-centralisé peut être envisagée afin d'échanger des informations ou de se connecter à un service particulier, la connexion à des infrastructures fixes se faisant de manière ponctuelle. Les réseaux spontanés quant à eux peuvent être vu comme des réseaux où le multi-saut n'est pas pris en compte : des terminaux mobiles s'échangent des messages à un saut de façon opportuniste lorsqu'ils sont à portée radio les uns des autres, la connectivité globale du réseau étant uniquement assurée par les mobiles le constituant. Il s'agit donc de réseaux auto-organisés à un saut. Ce modèle correspond à celui des *Pocket Switched Networks* (PSN) [HLC+06] qui se définissent par les utilisateurs autonomes qui les composent.

9. `http://kaa.citi.insa-lyon.fr/`

Le problème envisagé ici est celui de la confiance dans les communautés ouvertes qui constituent les réseaux spontanés car les mobiles qui composent ce réseau appartiennent à des individus eux-mêmes membres de groupes sociaux. Il s'agira donc de construire un modèle de communautés ouvertes de service [LHU03, YHC07] tenant compte des mouvements dynamiques de la communauté (naissance, partitionnement, jointure) ainsi que de l'évolution de ces communautés (taille, population). Il faudra également prendre en compte les aspects de sécurité pour ce type de réseau afin de garantir les propriétés classiques de sécurité comme l'authentification ou la confidentialité. En effet, nous souhaitons que notre modèle puisse garantir les propriétés usuelles de sécurité afin d'assurer une confiance minimale dans le système lui-même. Nous avons donc construit une solution où la prise de décision d'un échange repose sur la confiance en considérant que chaque entité se protégera d'éventuels voisins malicieux uniquement grâce à des informations locales dont il dispose. Notre modèle de gestion de la confiance a donc pour objectif de réduire l'incertitude provenant d'une information partielle. Il devra s'établir en fonction des services proposés et s'adapter dynamiquement à l'échelle de la communauté. Il privilégiera des modes de fonctionnement ne nécessitant aucune administration centrale et peu d'interventions de la part de l'utilisateur. Enfin, il facilitera et incitera la collaboration entre les mobiles.

1.2 Approche sociale de la confiance

La confiance est un terme à la fois commun auquel chacun peut donner une définition et très ambigüe tant il peut recouvrir des aspects très variés. Dans des versions technologiques, la confiance a été abordée pour le web, les réseaux pair-à-pair, dans le cadre des codes mobiles ou parfois dans des contextes très spécifiques, comme les applications médicales [GS00].

On peut définir d'un point de vue général la confiance comme un mécanisme de coordination des échanges (économiques ou non) en situation d'ignorance ou d'incertitude : c'est elle qui permet de prendre une décision malgré l'existence d'un risque. La confiance ne peut se définir que vis-à-vis d'une personne, qui peut être éventuellement morale. Cette notion se distingue ainsi de celles qui s'appliquent généralement aux techniques ou aux technologies comme la sécurité, la fiabilité ou la sûreté même si ces notions sont évidemment liées. Par exemple, d'après [Gam00], *"La confiance est la probabilité subjective par laquelle un individu Alice prévoit qu'un autre individu, Bob, exécute une action dont son bien-être dépend."*

Définition d'un modèle de confiance

De loin le plus répandu pour la communication inter-objets, le protocole Bluetooth [Blu01] est fondé sur le modèle de l'appariement : une information identique (un numéro PIN) doit être entrée manuellement dans chaque objet pour l'établissement d'un canal sécurisé. Cet appariement peut être temporaire (durée d'une session) ou définitif (réservé à l'appariement des objets d'une même personne). Le modèle n'est pas du tout adapté aux interactions spontanées car il nécessite l'intervention systématique du propriétaire, et ne possède pas de degré d'association au sens de niveaux de confiance requis pour accéder à différents services (ces autres aspects doivent être gérés en dehors du protocole) ou de remise en cause en cas de lien déclaré permanent.

Notre définition d'un modèle de confiance est un mécanisme technologique qui se substituerait au modèle d'appariement dans le but de diminuer la nécessité de l'intervention humaine et incorporant un mécanisme aussi automatique que possible pour la graduation de la confiance et la répudiation de liens préalablement établis. Remarquons que cette définition ne cherche pas à supprimer totalement l'intervention du propriétaire de l'objet, cette suppression n'est pas souhaitable dans notre vision pluri-disciplinaires. En effet, pour être socialement accepté,

il est primordial que le propriétaire de l'objet garde un certain contrôle sur celui-ci, ne serait-ce que parce que certaines situations ne peuvent être évaluées automatiquement. Enfin pour être socialement acceptable, le contrôle du propriétaire est également nécessaire parce qu'il reste responsable (socialement, économiquement ou juridiquement) des conséquences engendrées par les interactions qu'il suscite ou accepte. Enfin, dans le cas particulier des objets communicants, la confiance dans l'objet est directement liée à la confiance en son propriétaire, ce qui lui confère un rôle social.

On distingue en général deux types de confiance : la confiance a priori, dite également aveugle ou assurée, qui est acquise ex ante et la confiance décidée qui est le résultat d'un réel processus d'appréciation du risque qui passe par une gestion de celui-ci et par une prise de décision négative ou non. La première se passe d'une réelle évaluation du risque car on estime que, dans ce cas, la réalisation du risque est très improbable, que les inconvénients possibles sont minimes face aux avantages attendus ou que l'on n'a pas de réelle alternative. Ce modèle s'applique à des groupes fermés, mais bénéficie dans notre approche d'un mécanisme permettant de renforcer cette confiance pour se protéger contre certaines attaques internes aux groupes telles que l'usurpation d'identité provenant d'une attaque informatique ou tout simplement du vol d'un objet mal configuré permettant l'usage de celui-ci par un tiers. L'un des objectifs d'un système de confiance devrait être de rendre difficile le maintien de ce niveau de droit par un utilisateur non autorisé. Pour la seconde forme, il s'agit ici d'évaluer les avantages attendus ainsi que les inconvénients qui peuvent découler d'une prise de décision de confiance, cette décision pouvant être positive ou non. Toutes ses étapes sont alors parfaitement conscientes même si la décision prise peut conduire à la déception et au regret. Évidemment, si les dommages potentiels sont supérieurs aux avantages induits, il s'agira de prendre une décision négative et de ne pas choisir l'interaction. Elle s'applique aux relations à l'intérieur de groupes socialement très ouverts ou entre les groupes. Dans un tel contexte, il est communément admis que la relation de confiance n'est pas transitive. Par contre, une transitivité partielle peut être envisagée comme provenant de mécanismes de réputation ou de recommandation. Dans ce cas, ce n'est pas la confiance qui est transitive, mais l'information sur laquelle la décision de confiance se fonde. Cette transitivité permet d'accélérer la diffusion de l'information, ce qui améliore grandement la performance du système, tant que la quantité d'information fausse reste faible [MB05].

Dans les deux cas, on admet facilement que l'établissement de la confiance est fortement contextuel : elle dépend du service demandé, mais également du contexte dans lequel cette demande est faite. Ce contexte peut dépendre d'une position géographique, de la liste des objets actuellement à portée radio, de la liste des interactions qui ont eu lieu localement,... Il correspond parfaitement à un comportement social : vous prêtez les clés de votre voiture à votre professeur de tennis pour la déplacer lorsqu'elle gêne, cependant il n'est pas sûr que vous acceptiez cette requête si celle-ci est faite dans un lieu totalement différent de votre club de tennis.

Le modèle de confiance a pour objectif de réduire l'incertitude provenant d'informations partielles. Les modèles à base de réputation et de confiance ont été largement étudiés, le plus souvent dans le cadre de l'Internet. Dans ce dernier cas, le réseau bénéficie d'une caractéristique que ne possède pas les réseaux d'objets communicants : la grande capacité de communication - tout acteur est potentiellement atteignable - et la capacité importante de stockage des informations et de traitement de celles-ci. Dès lors, les statistiques peuvent venir au secours de ce type de modèle, en garantissant par exemple qu'il soit très difficile à un individu seul ou même à un petit groupe d'individus de fausser le système de confiance en influençant de manière significative la réputation envers certains membres du système.

Confiance et groupes sociaux

On retrouve bien dans cette courte introduction sur la confiance les deux objectifs d'un modèle de confiance qui sont de résoudre le problème d'accorder des droits - ce qui peut se résumer à accepter une requête ou à la rejeter -, en fonction de son identité et de l'identité du demandeur. Les solutions proposées pour l'Internet sont souvent caduques dans le contexte des objets communicants. Tout d'abord, nous nous trouvons d'emblée face à un ensemble de communautés et non pas a une collection de participants appartenant tous à une même communauté. Par communauté - ou domaine de sécurité, on entend ici un ensemble de couples objet/personne ayant une identité certifiée qui leur permet d'être authentifié par d'autres membres de la communauté. Une communauté pouvant éventuellement se résumer à un seul objet/personne (ou à une collection d'objets appartenant à la même personne). Dans [LHU03, LNAU03, GLNU05], les membres du projet KAA ont catégorisé les groupes sociaux en fonction du mode de régulation des échanges. Bien qu'il s'agisse de communautés fermées, un mécanisme de régulation de la confiance reste nécessaire comme nous l'avons évoqué. Pour résumer notre approche, quatre modèles peuvent être discriminés suivant deux variables : la *distance sociale* qui sépare deux individus qui désirent interagir - celle-ci pouvant être forte lorsque ces individus ont peu de relations sociales ou faible dans le cas de relations sociales suivies -, et le *degré de structuration* du groupe, qui définit les degrés de liberté dont disposent les acteurs pour interagir - un degré faible signifiant une très grande liberté d'actions alors qu'un degré fort implique de très fortes contraintes sociales.

Au sein de chacun de ces quatre modèles, les interactions sont, en théorie, réglées par un mécanisme dominant : le don pour un modèle de type famille (il faut entendre ici par famille tout groupe aux relations sociales fortes qui n'ont pas besoin d'autre motivation que ces liens pour interagir), l'autorité pour les groupes de type organisation (par autorité on entend un système de délégation de droit qui engendre des nécessités ou des obligations d'interactions), le prix pour des groupes de type marché (par prix on entend tout mécanisme possédant un système de valeur pour les interactions) et enfin la confiance pour les groupes de type réseaux (on entend confiance au sens commun du terme comprenant des notions telles que la compétence d'autrui, ses connaissances, sa position dans le réseau,...).

		Distance sociale	
		Forte	**Faible**
Degré de structuration	**Fort**	Organisation/autorité	Famille/don
	Faible	Marché/prix	Réseau/confiance

Notre modèle de confiance est fondé sur la notion d'historique commun entre deux objets désirant réaliser une interaction spontanée. Dans ce modèle, notre identité sociale se confond avec notre passé : nous sommes ce que nous avons fait. Pour cela, des informations sont enregistrées dans un historique d'interactions passées entre chaque mobile avec comme but de réduire l'incertitude sur la conduite des autres entités. Deux mobiles peuvent échanger des services si ils peuvent se faire confiance, c'est-à-dire si ils ont un nombre suffisant d'interactions passées avec d'autres entités. Chaque élément de l'historique sera donc un bout d'une confiance obligataire prouvée grâce à des outils cryptographiques. Le processus de prise de la décision de confiance sera donc fondé dans notre modèle sur les éléments d'historique et sur l'expérience propre de chaque entité. Nous créons ici un premier niveau de confiance : la confiance en l'interlocuteur.

L'architecture utilisée pour le modèle de confiance permet d'atteindre un deuxième niveau de confiance : la confiance dans le système. En effet, si l'on admet que notre architecture est impossible à violer - ou plus modestement que les barrières sont suffisantes -, alors nous créons une confiance envers le système. En effet, nous ne sommes pas forcément convaincus de prendre la

bonne décision ou que la politique de sécurité que nous utilisons soit la meilleure, mais nous savons qu'elle prend sa décision sur des données qui sont cryptographiquement prouvées. Pour cela, afin de remplacer un tiers de confiance qui ne peut pas être mis en place dans un réseau ambiant, nous utilisons des identités certifiées afin d'assurer de solides fondations cryptographiques à notre modèle. C'est pour cela que nous utilisons un historique cryptographique. Dans ce cas, chaque élément de l'historique d'une entité pourra être vérifié par une autre entité sous réserve d'une connaissance commune. L'architecture complète de notre modèle est décrite à la Section 1.3.

Il existe d'autres niveaux de confiance qui sont nécessaires pour qu'un système de confiance puisse passer à l'échelle. Une confiance de groupe est nécessaire, cette dernière étant également dotée de son propre mécanisme de régulation. Un individu doit avoir raisonnablement confiance dans les membres de son groupe. Il faut également construire un mécanisme permettant d'acquérir une confiance envers un groupe auquel on n'appartient pas afin que les différents groupes puissent interagir. Enfin, le dernier niveau serait la confiance dans le modèle par rapport à la société elle-même : qui me garantit une certaine sécurité en cas de dérapage du système ? On touche ici à une régulation de la confiance qui transcende le simple individu et qui est habituellement abordée par la régulation juridictionnelle et la mise en place d'un système de sanctions. Notre approche pour ce problème n'est pas de se substituer aux processus de régulation sociale qui existent, mais de proposer un système qui permet de s'y intégrer.

La régulation des systèmes de confiance face à la société va prendre de plus en plus d'ampleur dans un avenir proche : les consoles de jeux portables sont mises entre les mains d'enfants de plus en plus jeunes et sont maintenant dotées de capacité de communication, entre les consoles et avec des serveurs de l'Internet ! Un juriste vous dirait rapidement que, contrairement aux idées reçues, il n'y a pas de vide juridique lié à l'apparition de nouvelles technologies. Par contre, il peut résulter de l'apparition de nouvelles technologies une difficulté à constater l'infraction. Pour corriger ce problème, on peut influencer la technologie en y ajoutant des mécanismes facilitant l'enquête en cas de contestation. On peut également influencer les acteurs des systèmes eux-mêmes en introduisant des textes juridiques spécifiques, comme ceux imposants aux fournisseurs d'accès à Internet certaines obligations sur la traçabilité des connexions. Un modèle de confiance doit donc permettre une certaine traçabilité des interactions pour que l'utilisateur ait confiance dans le système. Ces traces, en étant cryptographiquement liées à ceux qui les ont générées, facilitent la contestation devant un système juridique. Nous avons pris ici l'exemple de la régulation juridique étatique, qui offre une protection aux citoyens d'un État, mais on peut l'appliquer également au sein de groupes se dotant de règles internes. Si l'on veut dépasser l'unique sanction qu'il est possible de prendre de façon autonome, c'est-à-dire le bannissement de la personne incriminée à tort ou à raison, le mécanisme de régulation de groupe, pour être socialement acceptable, doit, autant que possible, se fonder sur des faits établis et là encore, un historique cryptographiquement prouvable est utile. Il s'agit de mettre en place des mécanismes qui ne pourront pas être contestés ou qui pourront être vérifiés par des tiers. Dans le cas d'un groupe, il s'agit plus d'une régulation déontologique où un ensemble d'individus accepte un certain nombre de règles régulant leurs relations internes et/ou leurs relations avec d'autres groupes, auxquels s'ajoutent un mécanisme de sanction. Ces mécanismes nous entraîneraient trop loin par rapport à notre propos. On ne retiendra que le fait que l'architecture de sécurité pour objets communicants doit permettre de faciliter l'enquête sur les événements survenus dans le système, avec son pendant, le respect de la vie privée.

L'article 12 de la déclaration universelle des droits de l'homme de 1948 définit le respect de la vie privée : "Nul ne sera l'objet d'immixtions arbitraires dans sa vie privée, sa famille, son domicile ou sa correspondance, ni d'atteintes à son honneur et à sa réputation. Toute personne a droit à la protection de la loi contre de telles immixtions ou de telles atteintes."

Autant la loi française semble bien encadrée cette notion via la jurisprudence quand elle s'applique à la vie réelle dans des domaines comme la domiciliation, l'image, le secret médical ou la correspondance privée, autant l'utilisation d'internet ne permet pas les mêmes garanties. On peut citer ici l'exemple des réseaux sociaux et notamment de Facebook qui permet notamment à des employeurs de "tracer" les activités de leurs salariés.

Notons également que récemment, la France s'est illustrée par l'adoption d'une loi concernant le "droit à l'oubli numérique" permettant de simplifier et faciliter la suppression et la desindexation de données personnelles publiées sur Internet. Malheureusement, cette loi ne touche que les sites hébergés en France et donc peu de réseaux sociaux. Aux Etats-Unis où les données personnelles ont une valeur marchande, la solution passe souvent par des méthodes d'obfuscation.

1.3 Architecture et protocoles

Nous allons présenter les propriétés que nous avons jugées importantes pour notre modèle de confiance dans les réseaux spontanés. Comme la majorité des propositions existant dans la littérature, le protocole que nous avons développé comportera deux temps : un premier temps d'établissement de la confiance où les entités en présence décideront ou non d'établir un premier lien de confiance puis la gestion de cette confiance au cours du temps. Je ne donnerai pas ici l'état de l'art concernant les autres modèles existants du domaine, le lecteur peut se reporter à [GMU07, Cap04, QHC06] où à l'annexe B.

Notons tout d'abord que l'anglais, contrairement au français, fait une différence entre celui qui demande la confiance (dans notre cas, il s'agira d'un service) et qui en général est celui qui initie l'échange c'est le *trustor* et celui qui reçoit la confiance, le *trustee*, qui ici recevra la demande du *trustor*.

Principes du modèle de confiance retenu

Dans notre modèle (que nous appellerons modèle KAA du nom de notre projet) présenté dans [GMMU06, GMU07] (ce dernier article est présenté en annexe B), une entité existe si elle est équipée d'un package cryptographique minimum, qui la rendra compatible avec tout autre entité du modèle, c'est-à-dire d'autres objets qui acceptent implicitement le modèle. Ce package intègre le matériel cryptographique - fondé sur les courbes elliptiques - et des données initiales encore appelées *graine de confiance initiale* fournies par une entité appelée *station d'imprégnation*. Cette graine regroupe un ensemble de méthodes cryptographiques qui, une fois paramétrées, serviront de base à une politique de confiance. Ce paramétrage a lieu lors d'une phase d'initialisation dite ici d'imprégnation si on suit le vocable de [SA99]. Notre approche s'éloigne du modèle d'imprégnation original [SA99], par le fait qu'il ne crée pas un appariement fort entre un objet et son propriétaire, mais plutôt un contexte d'utilisation pour un utilisateur sur un objet particulier. Notre approche est d'éviter de donner une sémantique à une opération qu'il serait très difficile de garantir technologiquement sur une très large variété d'objets. Retenons simplement que nous faisons l'hypothèse que plusieurs imprégnations peuvent être appliquées au même objet, définissant alors plusieurs contextes d'utilisation et plusieurs identités. Chacune de ces stations définit un domaine de sécurité. Dans chacun de ces domaines, un objet a reçu de la station sa *graine de confiance initiale* construite à partir de la clé maître s propre à chaque station. Ainsi une même entité possédant plusieurs identités sur un même objet pourra choisir l'identité à utiliser selon le contexte. En reprenant l'exemple du club de tennis, une personne peut avoir une identité *Tennis* très différente de son identité *Profession*.

A partir du moment où il est imprégné, un objet peut initier une session de communication. Le protocole Common History Extraction (CHE) (décrit dans [GMMU06]) permet d'initier cette session et de la terminer. Si la session est acceptée, c'est que les deux entités en présence estiment, au vue de leur politique de sécurité, qu'elles peuvent se faire confiance pour cette interaction. Lors de l'ouverture d'une session, avant d'accepter de fournir ou d'utiliser des services, les objets en présence doivent construire un *germe de confiance* en regardant les entités communes qu'ils ont rencontrées auparavant. Pour prouver ces interactions, ils ont créé avec chacun des objets rencontrés auparavant un élément d'historique lié à leur identités qu'ils ont respectivement signés. A partir d'un historique vide, un objet enregistre toutes les interactions qu'il a faites avec d'autres entités. En enrichissant l'historique de chacun des objets, on facilitera ainsi les interactions spontanées futures. Si une entité possède plusieurs identités ou contextes d'utilisation, elle devra choisir dans quel contexte et avec quelle identité a lieu l'interaction.

Le protocole CHE (Common History Extraction) d'évaluation de l'historique commun En raison de la nature déconnectée du réseau que nous étudions, nous avons décidé d'utiliser la cryptographie fondée sur l'identité qui permet par construction de se passer de certificats.

Le matériel cryptographique fourni à chacun des objets du domaine par la station d'imprégnation est constitué d'une identité unique ID (par exemple une adresse IP ou un nom), d'une première paire de clé secrète/clé publique $(S_{\mathrm{ID}}, Q_{\mathrm{ID}})$ servant au chiffrement, d'une deuxième paire de clé secrète/clé publique $(S^S_{\mathrm{ID}}, Q^S_{\mathrm{ID}})$ servant à la signature et d'un ensemble de paramètres publics nécessaires aux calculs sur les courbes elliptiques et communs à toutes les stations d'imprégnations et à toutes les entités. Dans ce cadre, une entité mobile seule peut devenir son propre domaine de sécurité en s'imprégnant elle-même. Le seul impératif est que chaque entité mobile doit posséder les mêmes algorithmes cryptographiques : des fonctions de hachage, un algorithme de chiffrement/déchiffrement, un algorithme de signature, un protocole zero-knowledge et tous les paramètres publics. Les seules valeurs qu'un objet doit absolument garder secrètes sont ces deux clés privées : S_{ID} et S^S_{ID}.

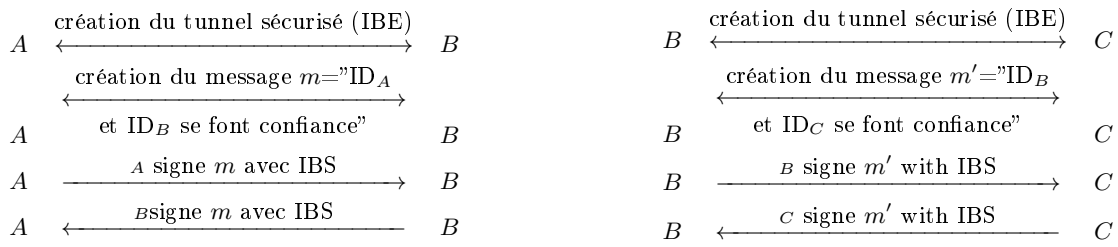
Nous avons décider d'utiliser IBE le schéma de chiffrement fondé sur l'identité de Boneh et Franklin [BF01] pour construire la première paire de clés dédiée au chiffrement de chacune des entités, pour chiffrer des messages mais aussi pour construire le canal sécurisé à authentification faible nécessaire à l'initialisation de notre protocole. Nous avons également utilisé IBS, le schéma de signature fondé sur l'identité proposé par Chen, Zhang et Kim en 2003 et décrit dans [CZK03].

La première étape de notre protocole se situe lorsque les deux nœuds Alice et Bob souhaitent interagir, une fois qu'ils ont déjà construit un lien de confiance selon l'un des trois cas suivants : ils se sont déjà rencontrés et n'ont plus qu'à reconstruire un lien de confiance plus récent et plus riche sémantiquement ; ils ont construit un lien de confiance à l'aide d'un nombre suffisant d'éléments communs de leur historique ou durant une phase d'initialisation entre eux ; ou, les utilisateurs de chacun des mobiles ont forcé à la main le début de l'interaction comme dans un système de type Bluetooth.

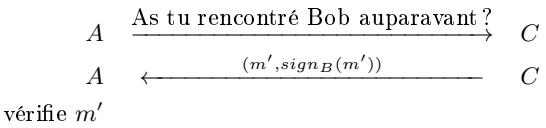
Une fois qu'Alice et Bob ont interagi, ils ont construit et signé, à travers un tunnel sécurisé grâce au protocole IBE, un message m de confiance réciproque. Alice publie dans la partie publique de son historique $(m, Q_B, Q^S_B, sign_B(m))$ tandis que Bob rend publique

$$(m, Q_A, Q^S_A, sign_A(m))$$

dans son propre historique. De la même manière, Bob et Charlie ont construit à travers un tunnel sécurisé le message commun m'.



Quand Alice rencontre Charlie pour la première fois, ils échangent la concaténation de toutes les clés publiques Q_{ID} qu'ils ont dans leur historique. Pour éviter les attaques de type déni de services, on suppose que la personne à qui l'on demande quelque chose est celle qui, en premier, fournit cette chaîne de caractères au demandeur. Une fois ce premier échange effectué, Alice et Charlie se rendent compte qu'ils ont tous les deux rencontré Bob auparavant et ils veulent se prouver l'un l'autre cette rencontre mutuelle. Charlie, tout d'abord, prouve à Alice que Bob lui fait confiance grâce à la possession du message m'.



Le même procédé est répété pour Alice. Ce protocole garantit toutes les propriétés cryptographiques usuelles : l'authenticité (comme Charlie connaît la clé publique de Bob, il peut authentifier sa signature et également l'identité d'Alice), l'intégrité est garantie par l'utilisation de fonctions de hachage dans le schéma IBS comme dans le cas classique du certificat, la confidentialité est garantie par le tunnel sécurisé. Ce dernier permet également de se prémunir contre les attaques de l'homme du milieu car l'authentification réciproque est faite par le lien de confiance qui unit les acteurs : le message m qui est connu seulement par les entités concernées et qui a au préalable été échangé de façon sûr par IBE (pour les détails concernant la résistance aux attaques classiques, le lecteur peut se référer à l'annexe B).

Notre vision de la confiance dans les réseaux spontanés est donc une relation pseudo-transitive qui contraste avec l'approche transitive du modèle Pretty Good Privacy (PGP) ([Fou99]). Cette approche est également celle utilisée dans [BMR04] ou [Cap04, QHC06]. Elle permet d'éviter qu'une collusion de nœuds détruise la confiance en un nœud particulier.

Nous avons également construit une version plus sécurisé de notre protocole nommée CHEWA pour "Common History Extraction With Authentication" qui permet une authentification forte afin de se prémunir contre la faiblesse inhérente d'IBE, le "key escrow drawback" décrite dans [BF01], à savoir, que la station d'imprégnation doit être honnête car elle peut déchiffrer tous les messages des entités qu'elle a imprégnées, la paire de clé secrète/ clé publique des objets étant construite à partir de la clé secrète de cette même station. Nous avons pour cela utilisé une certification off-line à deux niveaux fondée sur l'identité et proposé par C. Gentry dans [Gen03]. Les certificats sont vérifiés durant la première phase du protocole : la création du tunnel sécurisé via un algorithme de chiffrement/déchiffrement dédié à ce cas. Notons également qu'il est possible de rafraîchir les identités via l'utilisation d'une chaîne de MACs.

Les principaux paramètres de notre modèle sont donc le seuil p déterminant le nombre d'éléments d'historique communs que deux entités doivent posséder au minimum afin d'initier la

confiance, la taille maximum de l'historique lui-même H_{max} et la taille de la communauté considérée au départ.

1.4 Instanciation du modèle

Sémantique des éléments d'historiques et paramètres retenus pour le calcul de la note de confiance

Les éléments d'historique peuvent contenir un certain nombre de données sémantiques avant d'être signés par les deux parties : les identités des nœuds, la nature du lien et les termes de l'accord des deux parties, la date de création de l'élément, sa durée de validité ou éventuellement le lieu. Par exemple, un nœud A va stocker ses nœuds de confiance dans son historique H_A (pour chaque nœud de confiance B il y a un élément correspondant $h_A(B)$ dans l'historique H_A de A) et parallèlement, ce nœud stockera les nœuds qu'il considère malveillants dans BL_A sa *blacklist* (liste noire). Cette *blacklist* permet de tenir compte simplement et localement des nœuds qui n'inspirent pas confiance et avec qui A ne souhaite pas interagir. Nous noterons BL_{max} la taille maximale de la *blacklist*.

En plus des éléments d'historique eux-mêmes, nous avons décidé d'inclure les paramètres suivants dans le calcul de la note finale de confiance (le détail de la fonction est donnée en annexe B) :

- Le *taux d'interaction* : c'est le nombre moyen de demandes de services effectuées par un nœud par unité de temps. L'unité de temps sera définie par le protocole.
- *L'intensité* : elle est notée par $I_A(B)$ et représente le nombre de fois que B a interagi avec A.
- L'*utilité* de B pour A : elle est notée par $U_A(B)$ et représente le nombre de fois que $h_A(B)$ a été utilisé pour établir de nouvelles relations de confiance.
- Le *paramètre de communauté* $C(A,B)$: il indique si l'interaction a lieu entre deux membres de la même station d'imprégnation (ce paramètre vaut $C(A,B)$=1 dans ce cas, sinon vaut 0).

Ces informations peuvent être simplement stockées dans les éléments d'historique correspondants.

On peut également définir des communautés plus larges : une communauté peut ainsi être vue comme un ensemble de nœuds qui partagent un centre d'intérêt commun. Dans ce cas, le paramètre $C(A,B)$ peut être redéfini.

On peut également introduire une notion provenant de l'économie expérimentale : la réciprocité [Dub05, Dub06a]. Cette notion tente d'éviter la conduite *free-rider*, à savoir qu'un nœud particulier égoïste ne rend pas service aux nœuds qui lui ont rendu service auparavant. Pour tenter d'endiguer ce problème, on propose dans [GLM+06] la solution suivante :

1. Lors de la première étape, le *trustor* A a demandé au *trustee* B un service particulier.
2. Le *trustee* B a pu fournir le service au mobile A. Dans ce cas, l'élément d'historique créé sera signé par les deux parties en présence m='*B a fourni un service à A*'. Dans le cas où le service n'est pas fourni par B, l'algorithme s'arrête ici. Nous appelons cette propriété $sp_A(B)$='*preuve de service*' qui ne peut être vérifiée cryptographiquement tant qu'elle n'a pas été signée.
3. Le *trustor* A peut enrichir l'historique de B en signant et en lui envoyant le message m. Dans ce cas, nous appelons cette propriété la $tp_B(A)$='*preuve du trustor*'. Si le mobile A ne fournit pas cet élément d'historique, l'algorithme s'arrête là et B peut noter pour A '*aucune preuve du trustor*'.

4. Le mobile B a reçu du mobile A le $tp_B(A)$ et il devient réciproque s'il fournit au *trustor* A un élément d'historique signé. Nous appelons cette propriété $rp_A(B)$='*preuve de réciprocité*'. Naturellement, le mobile B n'est pas obligé de fournir cet élément et personne ne pourra vérifier cette non-réciprocité.

Il n'existe aucun mécanisme pouvant garantir que la génération de l'élément d'historique aura vraiment lieu après l'interaction. Si le *trustor* refuse de générer cet élément, le *trustee* ne peut être réciproque. Cependant, les réactions observées dans les jeux économiques expérimentaux montrent que le comportement humain est réciproque lorsqu'intervient la confiance. C'est ce même processus qui permet de préserver l'existence des réseaux pair-à-pair [Dub06b].

Recommandation et réputation

Le modèle KAA récuse l'usage d'un mécanisme de recommandation au sens classique du terme, la confiance étant considérée comme non transitive. En effet, le principe du modèle KAA est de ne fonder la politique que sur des éléments qui sont cryptographiquement prouvables. Dès lors, il est impossible de recevoir d'un nœud une *recommandation* pour un nœud que l'on n'a jamais rencontré, car il n'existerait pas de moyen simple et local pour prouver la sémantique de cette recommandation. Par contre, rien n'empêche deux personnes du modèle KAA de se générer mutuellement un élément d'historique sans qu'aucun service n'ait eu lieu : en donnant un élément d'historique sans compensation j'affirme à la communauté que je recommande les interactions avec cet objet et on peut prouver de qui vient la recommandation et à qui elle s'applique.

De même, le modèle KAA permet la gestion d'un mécanisme de réputation, localement calculée, dès lors qu'il se fonde sur des éléments prouvés. Le calcul de la valeur de réputation du nœud A envers le nœud B peut être réalisé à l'aide d'informations de deux niveaux :

- **Direct** On peut garder le souvenir des interactions passées entre A et B, ainsi que la valeur sémantique des éléments d'historique qui avaient été générés. Cette information pourrait être prouvée à un tiers qui a déjà rencontré B si les éléments d'historique étaient horodatés. Cependant, le protocole deviendrait lourd à mettre en oeuvre et surtout, des coalitions de nœuds adverses pourraient générer artificiellement ce type de réputation. Pour cela, l'élément d'historique n'indiquera que la qualité (*trustor*, réciproque ou les deux) de la suite d'interactions passées, sans autres informations numériques. Dans ce cas, le nœud A est sûr des informations enregistrées et peut en dériver une valeur de réputation pour B.
- **Indirect** Lorsque A interagit avec des nœuds autres que B, il procède à un échange de tout ou partie de l'historique avec son correspondant. Chaque fois que son correspondant cite B dans son historique, A reçoit également la sémantique de cette interaction (*trustor*, réciproque ou les deux). Il peut donc en tenir compte pour mettre à jour la réputation qu'il a envers B. Bien évidemment, il n'y a pas d'information sur la quantité d'interactions que le correspondant de A a eu avec B et ce bonus de réputation ne pourra compter qu'une seule fois pour chaque correspondant autre que B et qui présente B dans son historique.

On peut donc constater que la politique de confiance indirecte ne viole pas les fondements du modèle KAA : seules des informations que l'on peut prouver cryptographiquement sont utilisées. Il est, par exemple, impossible de créer une entrée dans la table de réputation pour un nœud que l'on n'a jamais vu auparavant ou même de modifier une entrée précédemment construite. De plus, la construction d'une telle table est clairement une incitation à la réciprocité : en étant réciproque, les nœuds laissent des traces de leur bonne volonté dans les tables des nœuds avec lesquels ils ont interagi et donc peuvent espérer atteindre une très bonne réputation auprès de beaucoup de nœuds du système.

1.5 Modélisation, simulation et implémentation de notre protocole

Dans [GMU07], nous modélisons la fonction donnant une note de confiance. Celle-ci dépend de tous les paramètres décrit précédemment (comme les *blacklists* ou les coefficients de communauté). Cependant, il est clair que les principaux paramètres de notre modèle reste le seuil p déterminant le nombre d'éléments d'historique communs que deux nœuds doivent posséder au minimum afin d'initier la confiance, la taille maximum de l'historique lui-même H_{max} et la taille de la communauté considérée au départ.

Nous présentons ici plusieurs approches que nous avons testées afin de démontrer la pertinence de notre modèle. Dans [GMMU06], nous avons présenté une approche probabiliste ainsi que des résultats de simulations concernant des interactions dans des graphes sociaux. Dans [GMU07] présenté en annexe B nous nous sommes intéressés à l'évolution de la phase de *bootstrap*, à la meilleure politique d'éviction des éléments d'historique et à l'impact de la *blacklist*. Nous présentons également les résultats d'implémentation sur Pocket PC donnés dans [Gal07].

1.5.1 Approche probabiliste

Nous considérons ici un historique de taille H_{max} en considérant une politique d'éviction LRU (Least Recently Used), une communauté comportant n nœuds et nous cherchons à estimer le nombre p de nœuds en commun dans l'historique. On suppose ici que les rencontres entre les nœuds sont aléatoires et ne dépendent pas de lois particulières de proximité.

On en déduit alors la probabilité que A et B qui appartiennent à la même communauté de n nœuds aient au plus p connaissances communes (en excluant p) : $\frac{1}{(\binom{n}{H_{max}})^2} \cdot \sum_{i=0}^{p-1} \binom{n}{i} \cdot \binom{n-i}{H_{max}-i} \cdot \binom{n-H_{max}}{H_{max}-i}$.

La probabilité P qu'ils aient alors au moins p éléments d'historique communs est donnée par :

$$P = Pr(A \cap B \geq p) \quad = \quad 1 - \frac{1}{(\binom{n}{H_{max}})^2} \cdot \sum_{i=0}^{p-1} \binom{n}{i} \cdot \binom{n-i}{H_{max}-i} \cdot \binom{n-H_{max}}{H_{max}-i}$$

Nous avons calculé la probabilité correspondante de succès et nous avons observé en nous appuyant sur le paradoxe des anniversaires que pour un groupe de taille n, si la taille maximale de l'historique est de l'ordre de $n/\ln(n)$ et que le nombre p est $\sqrt{n/\ln(n)}$ alors, la probabilité de succès de créer un lien de confiance est de plus de 50 %. Ainsi, par exemple si $n = 100$, $H_{max} = 22$ et $p = 5$, la probabilité de succès est de $56,6\%$ (avec $p = 3$, cette probabilité atteint 92%). On peut ainsi constater avec cet exemple que la taille de l'historique reste raisonnable pour une petite entité mobile et que le nombre p de vérifications à faire est également raisonnable.

Nous donnons dans le tableau suivant quelques exemples des valeurs de n, H_{max}, p et P :

n	$H_{max} = n/\ln(n)$	$p = \sqrt{n/(\ln(n)}$	P
100	22	5	56,6 %
200	38	7	61,9 %
500	81	9	94,1 %
1000	145	13	98,9 %
2000	264	17	99,99 %
5000	588	25	100 %
10000	1086	33	100 %

1.5.2 Résultats de simulation

Graphe d'interaction

Notre modèle étant dédié aux petits objets communiquants appartenant à des personnes physiques, le graphe d'interaction associé est un graphe social. Ces graphes ont été étudiés depuis longtemps par les sociologues et les mathématiciens [Mil67]. La première propriété d'un graphe social est l'effet petit monde. Cette propriété signifie que même dans un graphe social fortement géographique (avec une partie insulaire ou des barrières sociales), il existe des chemins de connexions courts. Plus récemment, des travaux dédiés ont insisté sur l'importance de l'organisation en *clusters* qui peut affecter la manière dont le graphe social est construit. Cette dernière propriété, très importante pour la simulation, est la distribution inégale des degrés. Afin d'étudier le paramètre p, nous avons utilisé un graphe aléatoire avec une distribution inégale des degrés [PL05b]. La séquence des degrés est obtenue à l'aide d'un générateur suivant une distribution en loi de puissance.

Le but de nos simulations est alors de déterminer des valeurs correctes du paramètre p pour une communauté donnée. Plus précisément, il s'agit de choisir la bonne valeur de p qui donne une grande probabilité d'interaction spontanée entre les membres de la communauté et une faible probabilité d'interaction avec des nœuds n'appartenant pas à la même communauté. L'approche empirique a pour ce calcul besoin de la connaissance de la distribution de degrés à l'intérieur et à l'extérieur de la communauté. On considère donc une communauté C de 30 nœuds interagissant dans un groupe social G de 100 nœuds au total. On suppose que les interactions sont plus fréquentes entre les nœuds de C qu'entre les nœuds de C et de $G\backslash C$ (de même pour $G\backslash C$). Dans nos simulations, nous avons pris en compte 4 paramètres de communauté : s la taille de la communauté, d_{min} et d_{max} les bornes de l'intervalle des degrés possibles des nœuds et α l'exposant de la fonction de distribution de la loi de puissance. Les paramètres pour C et G sont donnés par :

	Nodes	d_{min}	d_{min}	α
C	30	6	12	2.4
G	100	5	10	2.4

On étudie ensuite, en fonction de p, le nombre de nœuds de $G\backslash C$ qui serait directement inclus dans C considérant un historique de taille infini. Il faut que le paramètre p soit à la fois choisi pour empêcher que la communauté C ne soit complètement absorbée par le groupe G et pour que les échanges soient quand même possibles au sein de C. Nous avons obtenu les résultats suivants concernant la valeur de p :

p	3	4	5	6
Nb de nœuds inclus dans la communauté	15-20	3-5	0-1	0
Dans C, probabilité d'interactions spontanées	99.9%	99.98%	98.66%	92.84%

On peut voir ici le nombre de nœuds de $G\backslash C$ spontanément intégrés à la communauté C selon la valeur de p. Ainsi, en fonction des paramètres, la communauté peut être relativement ouverte (en choisissant une petite valeur de p) ou fermé (en choisissant une valeur de p plus élevée), dans ce cas l'intégration de membres extérieurs ne sera pas aussi facile. La deuxième ligne du tableau montre également que quelque soit la valeur de p, la communauté fonctionne de toute manière extrêmement bien avec une probabilité d'interactions spontanées supérieure à 90% (pour un calcul réalisé avec un historique de taille 15).

Phase de *Bootstrap* et politique d'éviction de l'historique

Comme décrit précédemment, la phase de *bootstrap* a lieu lorsque l'historique d'un nœud est vide et jusqu'à ce qu'il possède suffisamment d'éléments d'historique pour pouvoir interagir automatiquement avec les membres de sa communauté. Ainsi durant la phase initiale où les relations de confiance sont régies à la main par l'utilisateur, chaque entrée ajoutée aura une grande influence sur la suite des relations automatiques. Les résultats de simulations fournis en annexe B nous montrent que le nombre d'interactions forcées décroît rapidement et que la probabilité que les interactions soient encore forcées à la main après un temps borné tend vers 0.

Dans [GMU07], nous avons également étudié les différentes politiques possibles de d'éviction d'un élément d'historique lorsque ce dernier est plein. Nous avons considéré les politiques suivantes : FIFO (*First Input First Output*) et LUFO (*Less Used First Output*). Comme attendu, les résultats de simulations présentés en annexe B montre que le mode LUFO est le plus efficace de tous pour garder la cohésion des interactions régulières car ce mode prend en compte l'importance des éléments dans l'historique. De plus avec le mode LUFO des valeurs de p autour de 3 ou 4 sont raisonnables pour des communautés de 100 ou 200 nœuds.

1.5.3 Aspects d'implémentation

Dans sa thèse, S. Galice [Gal07] fournit des premiers résultats concernant l'implémentation du protocole CHE sur des PDAs en utilisant les algorithmes ECDH et ECDSA de la bibliothéque Miracl [Sco88] en lieu et place de ceux fondés sur le calcul de pairings bilinéaires initialement proposés dans CHE. Il montre que les temps de calcul de signature et de vérification de ces schémas cryptographiques (de l'ordre de moins d'une centaine de milli-seconde sur un PDA pour une courbe de taille 160 sur un corps premier) restent suffisament raisonnables pour être utilisés. Dans ce cas, avec un historique de taille 30, les deux parties en présence devront s'échanger des chaînes de 600 octets pour échanger leur historique ce qui reste également raisonnable.

Chapitre 2

Solutions de sécurité pour les réseaux de capteurs sans fil

2.1 Introduction

Les réseaux de capteurs sans fil (en anglais Wireless Sensor Networks WSN [DP10]) peuvent être vus comme un nouveau paradigme entre réseaux de surveillance d'infrastructure et réseaux ad-hoc [Toh02]. Ces réseaux sont composés de petites entités autonomes (mobiles ou non), appelées nœuds capteurs, chargées de collecter des données physiques le plus souvent environnementales à l'aide d'un capteur et de les transmettre via le médium radio à un ou plusieurs points de collecte (les puits). Cette transmission se fait le plus souvent en mode multi-saut. Les domaines d'applications actuels et potentiels des réseaux de capteurs sont nombreux. On peut citer par exemple les applications militaires, la domotique, la surveillance industrielle ou de phénomènes naturels [BTB04], les relevés de compteurs, etc.

Les capteurs eux-mêmes, de par leur taille et leur autonomie, définissent un certain nombre de contraintes : fonctionnant le plus souvent sur batterie, leur énergie est limitée ; comme ce sont de petits dispositifs, leur capacité de calculs et leur mémoire sont également limitées ; la communication radio étant sans fil, ils n'ont qu'une faible portée radio et un faible débit. A titre d'exemple, les capteurs Senslab [10] développés au laboratoire CITI, possèdent un microcontrôleur TI MSP430-1611 avec un processeur 16 bits cadencé à 8 MHz, 48 ko de ROM et 10 ko de RAM, une interface radio TI CC1101 ou CC2420 avec des fréquences respectives de 868 MHz et 2,4 GHz, un numéro de série unique DS2411 sur 6 octets, une mémoire flash externe ST M25P80 de 1 Mo et une batterie de deux piles.

Les réseaux de capteurs eux-mêmes possèdent un certain nombre de caractéristiques : comme le réseau est dynamique (il faut tenir compte de la mort ou de l'arrivée de capteurs), l'infrastructure n'est pas fixe ; le réseau a une forte densité ; on considère souvent la topologie comme étant aléatoire (on suppose par exemple que les capteurs ont été jetés d'un avion) et le routage est multi-saut en raison des faibles capacités de transmission de chacun des nœuds capteurs.

Dans le cadre de la thèse de W. Znaidi [Zna10] financée par la région Rhône-Alpes du projet ANR ARESA2 [11] et de la thèse de O. Erdene-Ochir financée par Orange Labs Grenoble, nous nous sommes particulièrement intéressés à la sécurité des réseaux de capteurs. En effet, par leur nature même (communications radio et multi-saut, compromission possible des nœuds [BBD06], etc.), ils sont vulnérables aux attaques classiques par compromission des nœuds déjà connues

10. voir `http://www.senslab.info/`
11. ANR VERSO (2010-2013) : `http://aresa2.minalogic.net/`

contre les réseaux ad-hoc [HBC01]. Parmi ces attaques, on peut citer l'attaque *wormhole* (trou de ver) [KW03], l'attaque *sinkhole* [KW03] (trou du puits), l'attaque *selective forwarding* [KW03], le *jamming* [WSS03, LvHD+05]) ou d'autres attaques spécifiques ([BBD06]).

Exigences de sécurité

De manière générique, on peut définir un attaquant dans un réseau de capteurs en fonction des caractéristiques suivantes :

- Son intention : l'attaquant est actif ou passif (il intervient ou non sur les données échangées dans le réseau).
- Sa localisation : il est interne ou externe (il est autorisé dans le réseau ou non).
- Sa capacité : il est fort ou ordinaire (il dispose (ou non) de ressources supérieures à celles d'un nœud).
- Sa mobilité : il est mobile ou non.

Comme dans les réseaux classiques et afin de se prémunir contre un certain nombre de vulnérabilités élémentaires, l'utilisation de primitives cryptographiques particulières permet de garantir les propriétés classiques de sécurité que sont l'authentification (légitimité d'un nœud), le contrôle d'accès, la confidentialité (lutte contre les écoutes passives), l'intégrité des données (lutte contre la modification du contenu des données), l'authenticité des données (lutte contre la modification de l'émetteur d'une donnée). Ces mécanismes permettent essentiellement de prévenir les attaques externes. Afin de se prémunir contre des attaques plus complexes (compromission de nœuds, *wormhole*, *sinkhole*, etc.) ou pour garantir la protection des données de la vie privée, d'autres propriétés (comme l'anonymat) ou des mécanismes plus sophistiqués (comme l'appel à des protocoles d'authentification forte ou à des protocoles dédiés) sont parfois requis.

Les propriétés précédentes peuvent être garanties par l'utilisation de la cryptographie symétrique ou asymétrique. Dans le cas de la cryptographie asymétrique, le chiffrement des messages à destination du puits se fera à l'aide de la clé publique du puits simplifiant au maximum la phase de distribution des clés même si cette forme de cryptographie nécessite beaucoup de calculs et donc requière beaucoup d'énergie. Si c'est la cryptographie symétrique qui est privilégiée car elle est plus efficace en termes de calculs et donc de dépense énergétique, il est cependant nécessaire de définir des mécanismes de prédistribution de clés dédiés afin que deux nœuds voisins puissent communiquer ensemble de façon sûre (en supposant que le voisinage d'un nœud n'est pas connu à l'avance). Beaucoup de solutions ont été proposées pour remédier à ce problème. On peut citer ici le mécanisme probabiliste proposé dans [EG02] où chaque nœud est préchargé avec un anneau de clé, un sous-ensemble de clés pris dans un ensemble de clés communes plus large. Cet anneau est construit de telle manière que la probabilité que deux nœuds partagent au moins une clé commune soit élevée. Dans [CA05, LN03], deux mécanismes t-secure sont également proposés (i.e. au moins $t+1$ nœuds doivent être compromis afin de compromettre toutes les clés du réseau). Ces schémas sont très efficaces et semblent être à l'heure actuelle les meilleures solutions à utiliser. Avec W. Znaidi, dans [ZM10], nous avons également proposé un mécanisme de gestion de clés pour les réseaux de capteurs fondé sur le même mécanisme que celui présenté dans [LN03], c'est-à-dire l'utilisation de polynômes bivariés symétriques [BSV+98], sauf que dans notre cas, nous utilisons des polynômes trivariés bi-symétriques. Ceci permet de prendre en compte l'identité de chacun des nœuds et d'éventuels ajouts ou suppressions de nœuds via des identités liées ensemble par des doubles chaînes de hachage. Une chaîne de hachage est définie comme $x_0, \cdots, x_n$ avec $x_i = H(x_{i+1})$ où H est une fonction de hachage à sens unique, les x_i étant révélés dans l'ordre inverse de leur construction. L'idée de départ était de proposer un mécanisme permettant de prendre en compte une authentification des identités dans les deux sens entre les capteurs plus

anciens et les capteurs plus jeunes lorsque les déploiements sont échelonnés dans le temps et que l'on considère la mort de certains capteurs. Le mécanisme initial permettant de tenir compte d'une authentification descendante entre les capteurs les plus anciens et les capteurs les plus jeunes avait été décrit dans [BLM07].

Ainsi, les mécanismes de prédistribution de clés sont à présent des éléments bien maîtrisés de la littérature et peuvent être utilisés sans risque. On peut donc à l'aide des mécanismes de prédistribution de clés, de schémas de chiffrement symétriques, de fonctions de hachage et d'algorithmes MAC (Message Authentication Codes) sécuriser également les mécanismes de routage, comme proposé dans [HPJ05] qui présente une version sécurisée ARIADNE du protocole DSR [NV09]. Ce protocole se fonde sur le mécanisme TESLA [PCTS02] qui utilise l'asymétrie temporelle fournie par des chaînes de MAC ou de hachage pour construire son authentification.

De notre côté et afin de prouver le bien fondé de l'utilisation de la cryptographie symétrique, avec Nicolas Fournel et Stéphane Ubéda dans [FMU07], nous avons comparé les temps de calculs des différents chiffrements à flot finalistes software de la deuxième phase de la compétition eStream [12] sur un capteur dédié. Ce capteur est composé d'un ARM922T (plus précisément, un Altera Excalibur EPXA10 qui intègre un FPGA) et des périphériques habituels. Il a trois niveaux de mémoires : deux mémoires caches séparées de 8kO, deux mémoires scratch-pad de 256 kO et 128 kO (qui ne sont pas utilisées dans nos tests) et une SDRAM de 128MO. Les tests de performance effectués ont montré tout d'abord que même sur des environnements contraints, les chiffrements à flot restent très efficaces, SNOW v2 [EJ02] en tête et que les performances du Key/IV setup de Sosemanuk [BBC+08a] sont dégradées en raison de la taille du code, trop importante dans le cas de ce capteur.

2.2 Quelques propositions de protocoles de sécurité dans les réseaux de capteurs sans fil

Dans cette section, je m'attacherai particulièrement à présenter les solutions algorithmiques que nous avons développées pour détecter et empêcher des attaques particulières dans les réseaux de capteurs sans fil. Je définirai donc dans un premier temps les attaques particulières contre lesquelles nous avons proposé des mécanismes de défense avant de développer les solutions proposées.

2.2.1 Quelques attaques dans les réseaux de capteurs sans fil

Dans [ZMB08a], avec W. Znaidi et J.-P. Babau, nous avons construit et présenté une ontologie des attaques connues contre les réseaux de capteurs. Cette ontologie, à partir d'une étude exhaustive des attaques existantes et des solutions proposées dans la littérature, est définie selon les notions utilisées dans [BLU07] et inspirées de la théorie de l'action où une action est décomposée en une intention, un mouvement et des objets. Ainsi pour modéliser une attaque au sein d'un réseau de capteurs, on la définit en fonction des quatre concepts : intention, mouvement, cible, résultat.

Je présenterai ici trois attaques classiques particulières de la couche routage : le *selective forwarding*, l'attaque *wormhole* et l'attaque par réplication.

12. voir `http://www.ecrypt.eu.org/stream/phase2list.html`

Selective forwarding et *sinkhole* (trou du puits)

Dans une attaque de type *selective forwarding* [KW03], les nœuds malicieux transmettent la plupart des messages mais en jettent également de façon sélective, le réseau perdant ainsi une partie de l'information. Un exemple d'une telle attaque est l'attaque par trou noir où l'attaquant ne retransmet jamais aucune des données qu'il reçoit. Les nœuds les plus malveillants sont bien sûr ceux qui font du *selective forwarding* près du puits. Le but de cette attaque est donc d'altérer les données et l'information transmises par le réseau. Cette attaque ne requiert aucune capacité technique particulière et peut être exécutée par n'importe quel nœud qui participe au mécanisme de routage. La cible est généralement le puits mais peut aussi être un simple nœud. Le résultat peut conduire à une perte complète de toutes les données si tous les paquets sont altérés.

Il existe plusieurs méthodes permettant de se prémunir contre ce type d'attaques comme l'utilisation de routage multi-chemin [GGSE01] avec un choix aléatoire du prochain saut vers la destination ou l'utilisation du codage réseau [ACLY00] (voir également la Section 2.4)

Une version plus dévastatrice du *selective forwarding* est l'attaque *sinkhole* (ou trou du puits) [KW03]. Ici, les nœuds essayent tout d'abord d'attirer vers eux une grande partie du trafic en se rendant très attractifs pour la métrique de routage. Ces nœuds, une fois leur forfait accompli, sont donc des nœuds qui vont apparaître aux yeux des autres comme des voisins du puits voire le puits lui-même et vont donc faire converger tout le trafic grâce à cette fausse topologie. Il leur suffit alors de jeter les paquets qu'ils routent par *selective forwarding* pour créer un trou du puits et empêcher le réseau de fournir des données. En général, cette attaque est lancée d'un ordinateur portable ou d'un PDA afin de convaincre les autres nœuds de la supériorité de l'attaquant. Elle exploite le fait que les identités et les liens ne sont pas vérifiés et elle a pour cible les services fournis par le réseau.

Une solution pour se prémunir contre cette attaque est l'utilisation de protocoles de routage qui vérifient la viabilité bidirectionnelle d'un itinéraire avec des reconnaissances bout en bout au niveau de la latence et de la qualité de l'information [KW03].

Attaque *wormhole*

Une attaque *wormhole* [KW03] (voir Fig. 2.1) utilise une connexion hors bande, appelée aussi tunnel *wormhole* entre deux nœuds physiquement éloignés. Ce tunnel peut être installé grâce à une radio sophistiquée qui possède une portée et une puissance bien meilleures que celles des autres nœuds ou via une connexion filaire reliant les deux bouts du tunnel. Dans la pratique, cette attaque devient intéressante quand les nœuds sont physiquement éloignés (une dizaine de sauts). Cette attaque permet de créer beaucoup de faux liens et de fausses routes en faisant croire à des nœuds qu'ils sont peu éloignés afin de dérouter une partie du trafic via ce canal malicieux. Le but de l'attaquant est donc ici de créer une fausse topologie logique voire de détourner une grande partie du trafic permettant ensuite de lancer d'autres types d'attaques comme des attaques par *selective forwarding* ou des déni de services.

Beaucoup de mécanismes ont été proposés pour se prémunir contre cette attaque. Nous en présentons deux ici. Dans [HPJ04], Hu et al. proposent un schéma de "paquets à laisse" (i.e. packet leashes) pour détecter des attaques *wormhole*. L'idée principale ici est de rajouter de l'information (soit géographique, soit temporelle) aux paquets échangés entre les nœuds afin de limiter la distance ou le temps maximal de parcours d'un paquet. Les paquets "laisses" géographiques se basent sur l'ajout de la localisation des nœuds dans le paquet, ce qui permettra au nœud receveur d'estimer une borne limite sur la distance qui le sépare du nœud expéditeur. Le deuxième type de paquet "laisse" est temporel, et il se base sur le rajout d'un temps dans le paquet afin de limiter

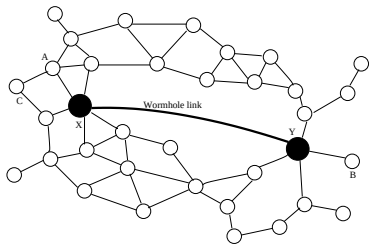


FIGURE 2.1 – Exemple d'une attaque trou de ver. **X** et **Y** sont les deux nœuds extrémités du tunnel de l'attaque trou de ver.

la durée de vie du paquet. Ce type de paquets "laisses" permet donc d'empêcher l'installation d'un tunnel *wormhole* dans le réseau. Cependant, cette proposition nécessite l'utilisation de coordonnées sûres via le GPS par exemple ou d'une synchronisation précise des horloges entre les nœuds. Ces deux mécanismes génèrent donc un surcoût important pour le réseau.

Dans [SPR+09], les auteurs proposent pour détecter les attaques *wormhole* d'estimer la distance entre un nœud et chacun de ses voisins. Cette estimation se fait par échange de messages simultanés sur une radio à ultrasons. Ensuite, les nœuds échangent entre voisins les distances qu'ils ont calculées et via des calculs trigonométriques, ils détectent la présence de *wormholes* lorsque des inconsistances de distances sont observées. L'inconvénient de cette approche est que chaque nœud doit être équipé d'une seconde radio à ultrasons, permettant l'estimation des distances entre les nœuds voisins.

Attaque par réplication de nœuds

L'attaque par réplication de nœuds, comme son nom l'indique, consiste à insérer des copies identiques de nœuds déjà capturés dans le réseau afin de créer une inconsistance. Même si un seul nœud est capturé, un attaquant peut en construire plusieurs copies et les insérer. L'attaque par réplication est une variante de l'attaque Sybille [NSSP04]. Dans une attaque Sybille, un nœud particulier possède et utilise plusieurs identités tandis que dans une attaque par réplication, une même identité est utilisée par plusieurs nœuds. Cette attaque lorsqu'elle n'est pas détectée, permet ensuite d'en lancer d'autres dans le réseau comme l'injection de fausses données distribuées, etc.

Les mécanismes permettant de détecter cette attaque sont de deux types : soit centralisés [NSSP04], soit décentralisés. La proposition décentralisée fondatrice est celle de Parno et al. [PPG05]. Les auteurs proposent deux algorithmes permettant de détecter l'attaque par réplication de nœuds : Randomized Multicast (RM) et Line Selected Multicast (LSM). Tous deux fonctionnent sur le même principe : la position de chaque nœud est diffusée vers un ensemble de nœuds témoins qui les enregistrent, et quand un nœud détecte une collision dans ces informations de localisation, il annonce la détection d'une attaque par réplication. Tous deux reposent également sur les mêmes hypothèses : d'une part, chaque nœud doit connaître sa position géographique ou une information de localisation, et d'autre part chaque nœud est capable de faire des opérations de cryptographie à clé publique permettant de garantir le chiffrement et la signature (tous les messages échangés entre les nœuds dans ces protocoles sont chiffrés, authentifiés et signés). Dans l'algorithme RM, chaque nœud α envoie sa position de localisation à ses d voisins directs. Ensuite, en utilisant un mécanisme de routage, chaque voisin diffuse avec une probabilité p cette information relative au nœud α à g nœuds choisis aléatoirement, g et p étant des paramètres

du protocole. Tous les nœuds témoins choisis par tous les voisins directs du nœud α constituent l'ensemble des témoins du nœud α. Si on fixe $p \cdot d \cdot g \simeq \sqrt{n}$ (où n est la taille de réseau et d le degré moyen), chaque nœud aura de l'ordre de $\mathcal{O}(\sqrt{n})$ nœuds témoins. Ainsi, par le paradoxe des anniversaires, deux nœuds répliqués auront un témoin commun avec une probabilité élevée (par exemple, les valeurs $d = 20$, $g = 100$ et $p = 0.05$ donnent une probabilité de détection de l'attaque supérieure à 95%). Lorsqu'un nœud témoin détecte une collision, il diffuse un message d'alerte à tout le réseau. L'inconvénient majeur de cette méthode est son coût en communication élevé : en considérant qu'envoyer un message vers un nœud témoin nécessite $\mathcal{O}(\sqrt{n})$ messages et que chaque nœud contacte $\mathcal{O}(\sqrt{n})$ témoins, le coût global de cet algorithme dans tout le réseau reste de l'ordre de $\mathcal{O}(n^2)$.

Afin de pallier à ce problème, les auteurs proposent dans le même article un deuxième algorithme : LSM. Ici, l'ensemble des témoins d'un nœud α est constitué par g témoins choisis aléatoirement mais aussi par les nœuds intermédiaires qui routent les paquets de localisation des voisins de α vers les nœuds témoins. Chaque nœud intermédiaire enregistre l'information de localisation, construisant ainsi une ligne témoin entre un voisin du nœud α et un nœud témoin. Ainsi, un nœud situé à l'intersection de deux lignes provenant de deux endroits différents du réseau peut détecter une réplication, la détection dans LSM utilisant le même principe de messages d'alerte que dans RM. Le coût total de communication pour LSM est donc de l'ordre de $\mathcal{O}(n\sqrt{n})$ et le coût en mémoire pour chaque nœud est égal à $\mathcal{O}(\sqrt{n})$ informations de localisation à enregistrer.

2.2.2 Proposition d'un algorithme de détection de l'attaque *wormhole*

Avec W. Znaidi et J.-P. Babau, nous avons proposé à PIMRC 2008 [ZMB08b] un algorithme de détection de l'attaque *wormhole*. Cet article est également présenté dans l'Annexe B. Ce mécanisme se base uniquement sur l'information locale disponible sur chaque nœud (la liste des voisinages à 1 et 2-saut), il ne nécessite aucun matériel spécifique et il est exécuté localement par chacun des nœuds pour détecter ou non la présence d'une attaque *wormhole*. Le mécanisme proposé exploite la topologie faussée induite par la présence d'un *wormhole*.

Coefficient de clustering des arêtes

Ce mécanisme se fonde sur le coefficient de clustering des arêtes, appelé en anglais edge-clustering coefficient ou ECC, tel que défini dans [RCC+04]. Ce coefficient représente la proportion de liens existants vraiment dans le réseau parmi ceux pouvant exister (ces liens peuvent être multi-saut). C'est une mesure de la densité de connectivité du réseau. Ce coefficient se calcule comme le rapport entre le nombre de structures géométriques (triangles, carrés, etc.) existants sur une arête et le nombre de structures géométriques pouvant exister sur cette même arête. Plus formellement, si on représente un réseau de capteurs statique comme un graphe $G = (V, E)$, où $V = \{v_1, \cdots, v_n\}$ est l'ensemble des sommets et $E = \{e_{ij}\}$ l'ensemble des arêtes où e_{ij} connecte les sommets v_i et v_j (en supposant les liens bidirectionnels, i.e. $e_{ij} = e_{ji}$). On définit le voisinage $V_k(a)$ à k-sauts d'un sommet a comme étant l'ensemble de ses voisins à k sauts, le degré associé étant $d_k(a) = |V_k(a)|$, le nombre de nœuds de $V_k(a)$. Le coefficient de clustering des arêtes à un saut $C_{i,j}^{(3)}$ entre les sommets i et j est alors défini comme le nombre de triangles qui inclut l'arête i-j divisé par le nombre de triangles qui pourrait inclure cette arête : $C_{i,j}^{(3)} = \frac{CS_{i,j}^{(3)}}{\min(d_1(i)-1, d_1(j)-1)}$ où $CS_{i,j}^{(3)}$ est le nombre de triangles construits sur l'arête i-j et $\min(d_1(i) - 1, d_1(j) - 1)$ est le nombre maximal de triangles potentiels.

Cette approche est généralisée à toute structure géométrique contenant plus de sommets. On parle alors du coefficient de clustering des arêtes d'ordre g : $C_{i,j}^{(g)} = \frac{CS_{i,j}^{(g)}}{s_{i,j}^{(g)}}$ où $CS_{ij}^{(g)}$ est le nombre de structures cycliques d'ordre g incluant l'arête i-j, $s_{ij}^{(g)}$ est le nombre de toutes les structures cycliques potentielles d'ordre g pouvant être construites entre i et j.

Définition d'un *wormhole* dans ce contexte

Si on cherche à définir ce qu'est un *wormhole* par rapport à son voisinage, on se rend compte qu'il s'agit d'un lien virtuel à 1 saut existant entre deux nœuds X et Y, les extrémités du *wormhole*. Normalement, dans un réseau suffisamment dense, deux nœuds réellement voisins à 1 saut vont avoir des voisins communs ce qui n'est pas le cas pour un *wormhole*. Ainsi, X et Y sont un lien *wormhole* si $((V_1(X) \setminus Y) \cap (V_1(Y) \setminus X) = \emptyset$ où $V_1(X) \setminus Y$ est le voisinage à 1 saut de X excluant Y. Exprimé en utilisant le numérateur du coefficient de clustering des arêtes, cela signifie que $CS_{X,Y}^{(3)} = 0$, c'est-à-dire qu'il n'y a pas de triangle réellement partagé entre X et Y. On obtient donc une condition nécessaire pour un lien *wormhole*.

La question devient alors comment un nœud a voisin d'une extrémité du *wormhole* (disons X) peut détecter celui-ci ? a peut supposer que son voisin à 1 saut X est un nœud *wormhole* si il ne peut pas atteindre Y, déclaré comme voisin de X par un autre moyen que X. En d'autre termes, cela signifie que a suppose que X est un nœud *wormhole* si le seul triangle qui peut être construit entre a et Y passe par X (en considérant une arête virtuelle entre a et Y) et si les seuls carrés qu'il peut construire sont $(a, X, \cdot, Y)$ et $(a, \cdot, X, Y)$. Cela signifie que le nombre de structures cycliques d'ordre 3 et 4 entre a et Y excluant X est nul. Si on note $CS_{i,j\setminus X}^{(g)}$ le nombre de structures cycliques d'ordre g excluant X, alors $CS_{a,Y\setminus X}^{(3)} = 0$ et $CS_{a,Y\setminus X}^{(4)} = 0$. Evidemment, la condition utilisée ci-dessus est seulement nécessaire, elle peut donc induire des faux positifs, i.e. des nœuds "isolés" qui sont déclarés comme étant des nœuds *wormhole* alors qu'ils ne le sont pas (on peut notamment citer les nœuds des bords comme faux positifs potentiels).

Algorithme de détection d'une attaque *wormhole*

Nous avons donc construit notre algorithme de détection de l'attaque *wormhole* en trois phases :

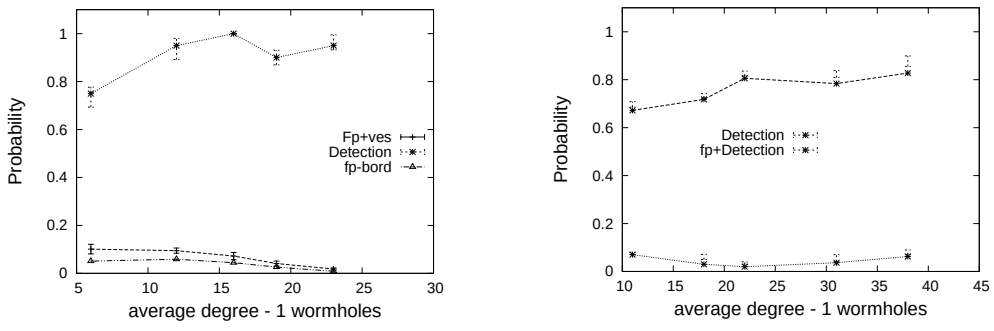
- **Découverte du voisinage :** Chaque nœud i, à l'aide de paquets Hello, détermine $V_1(i)$ et $V_2(i)$ et reçoit $V_1(v)$ et $V_2(v)$ pour chaque v dans $V_1(i)$.
- **Calcul de CS :** pour chaque j dans $V_1(i)$, i calcule $CS_{i,k\setminus j}^{(3)}$ pour chaque $k \in V_1(j)$. Si cette valeur est nulle, i calcule ensuite la valeur $CS_{i,k\setminus j}^{(4)}$. i déclare j comme un nœud soupçonné si cette dernière valeur est nulle aussi. Dans ce cas, le nœud j est ajouté à une liste rouge.
- **phase d'isolation :** Quand le nœud i insère le nœud j dans sa liste rouge, il diffuse un message d'alerte contenant l'identité du nœud j. Chaque nœud de $V_{1,2}(j)$ recevant cette alerte, ajoute j à sa liste rouge et incrémente le compteur relatif au nœud j. Quand un nœud w reçoit suffisamment de messages d'alertes (notons ce nombre par N_{am}) supérieur à un seuil donné T_{am} (par exemple $T_{am} = d_1(w)/3$), w envoie un message de détection de l'attaque *wormhole* à tous ses voisins pour isoler le nœud j en question du réseau et ainsi limiter les dommages provoqués par l'attaque *wormhole*. Ici c'est un mécanisme de vote classique qui est utilisé.

Comme présenté en Annexe B, les résultats de simulations réalisées à l'aide du simulateur WSNet [HCG08] en utilisant un modèle de propagation à disque (UDG) idéal et des couches

physique et MAC de type IEEE 802.11, montrent que, pour des graphes aléatoires, le taux de faux positifs passe en dessous de 10% tandis que le taux de détection passe au dessus de 90 % dès que le degré moyen est supérieur à 6 en considérant que $T_{am} = d_1(w)/2$ pour tout nœud w. Cette valeur de seuil est d'ailleurs le meilleur compromis entre un taux de faux positifs limité et une bonne probabilité de détection.

Extensions de ces travaux

Dans sa thèse [Zna10], W. Znaidi présente également ce même algorithme intégré à l'algorithme de localisation logique QLoP [HV08]. Le principe de cet algorithme est de classer l'ensemble des voisins d'un nœud en fonction de leur proximité géographique réelle. Cette proximité est calculée en tenant compte du nombre de voisins communs à deux nœuds. Ainsi, deux nœuds ayant beaucoup de voisins communs, se classeront respectivement dans leur 1-voisinage logique et ainsi de suite (QLoP définit 3 classes : le 1-voisinage logique, le 2-voisinage logique et le 3-voisinage logique). Il s'agira donc ici de remplacer la phase de découverte du voisinage de notre algorithme par une exécution du protocole QLoP afin de classer les voisins d'un nœud en fonction de leur proximité géographique réelle. Dans ce cas, notre algorithme ne calcule les coefficients CS que pour les voisins des classes 2 et 3 (un *wormhole* ne pouvant pas appartenir à la classe 1). En utilisant QLoP et le simulateur WSNet, nous avons pu constater que les résultats de simulations obtenus dans le cas d'un modèle de propagation réel (en présence de *pathloss* et de *shadowing* comme implémenté dans WSNet) pour notre algorithme étaient très bons (moins de 10% de faux positifs et plus de 90% de détection pour des degrés supérieurs à 8 et un lien *wormhole*, voir Fig. 2.2a) même en présence de plusieurs *wormholes*. W. Znaidi a également implémenté dans WSNet la solution proposée dans [SPR+09], cette solution dans un modèle de propagation réel donne des taux de faux positifs très faibles mais un moins bon taux de détection que notre propre solution (voir également Fig. 2.2b).



(a) (b)

FIGURE 2.2 – Probabilité de détection de l'attaque wormhole en environnement réel pour un graphe aléatoire (avec taux de faux positifs et taux de faux positifs excluant les nœuds du bords) : 2.2a pour notre protocole avec QLoP ; 2.2b pour le protocole proposé dans [SPR+09].

Dans le cadre du projet ARESA2, Chérifa Boucetta durant son stage d'ingénieur puis de MASTER, co-encadrée par M. Ali Kaafar et moi-même, a regardé comment se comportaient d'une part notre solution et d'autre part la solution proposée dans [SPR+09] lorsqu'on regardait des attaques plus complexes, c'est-à-dire des attaquants qui à la fois créent des *wormholes* et dont les voisins (ou eux-mêmes) mentent sur leur voisinage dans un même temps. Il s'est avéré que les simulations ont montré que notre algorithme résistait mieux que la solution de [SPR+09] même

si les résultats ne sont pas toujours concluants. Nous avons également commencé à proposer une amélioration de notre algorithme en présence de ce type d'attaquants via l'intégration d'un mécanisme de vote pour détecter les menteurs. Ces résultats sont publiés dans [BKM10].

2.2.3 Proposition d'un algorithme de détection de l'attaque par réplication de nœuds

Avec W. Znaidi et S. Ubéda, nous avons proposé à PIMRC 2009 [ZMU09] un mécanisme de détection de l'attaque par réplication de nœuds.

Hypothèses de travail et présentation de notre protocole

Ce mécanisme considère que le réseau de capteurs est hiérarchisé, c'est-à-dire que des clusters, constitués d'un chef (clusterhead) et de ses membres, ont été construits dans le réseau à l'aide d'un algorithme de clustering (dans notre implémentation il s'agit de l'algorithme LNCA [XV07] mais n'importe quel autre algorithme peut être utilisé). Dans ce cas, les clusterheads sont reliés entre eux via des nœuds passerelles afin de former un "backbone" permettant une communication simplifiée vers la ou les stations de base. Nous supposons également comme dans [PPG05] que des mécanismes cryptographiques permettant de garantir le chiffrement et la signature peuvent être utilisés.

Notre proposition est composée de trois étapes : une phase de prédistribution où chaque nœud reçoit le matériel cryptographique nécessaire, une fonction de hachage $h(\cdot)$ particulière dédiée au calcul des filtres de Bloom [Blo70] et une identité unique ; une phase de construction de l'architecture hiérarchisée via l'utilisation d'un algorithme de clustering ; et enfin la phase de détection des nœuds répliqués proprement dite. Cette dernière étape fonctionne de la manière suivante entre deux clusterheads particuliers CH_l et CH_i, le processus étant répété pour tous les couples de clusterheads (voir aussi Fig. 2.3) :

(1) le clusterhead CH_l construit la liste des identités de tous les membres id_j présents dans le cluster l, $S_l = \cup_{id_j \in CH_l} id_j$ en incluant sa propre identité.

(2) CH_l construit le filtre de Bloom BF_l relatif à l'ensemble S_l.

(3) il envoie au nœud CH_i le message M_l : $M_l = (E_{ke_{CH_l}}(BF_l)||Sig_{ks_{CH_l}}(BF_l))$ où ke_{CH_l} est la clé de chiffrement de CH_l et ks_{CH_l} sa clé de signature.

(4) CH_i, qui reçoit le message M_l, vérifie $Sig_{ks_{CH_l}}(BF_l)$ et déchiffre ensuite $E_{ke_{CH_l}}(BF_l)$ pour extraire le filtre BF_l.

(5) CH_i demande à un (ou plusieurs) nœud particulier id_r présent dans S_l (autre que CH_l) de construire de nouveau le filtre de Bloom du cluster l. id_r renvoie au nœud CH_i la valeur du nouveau filtre de Bloom BF'_l. CH_i vérifie si $BF'_l = BF_l$. Si oui, le filtre de Bloom est accepté et on passe à l'étape (6). Sinon, un message d'alerte est envoyé aux autres clusterheads qui vont commencer leur propre vérification pour le cluster l. Notons que pour trouver un candidat valide id_r, soit CH_i connaît déjà un nœud valide id_r, soit il effectue une recherche exhaustive sur BF_l en testant l'appartenance à BF_l d'identités Ids générées aléatoirement. (Cette étape est nécessaire afin de détecter si un nœud clusterhead est dupliqué ou non).

(6) A l'aide de sa propre liste de membres $S_i = \cup_{id'_j \in CH_i} id'_j$, le clusterhead CH_i vérifie si chaque identité id'_j appartient ou non au filtre BF_l. Si oui, cela signifie qu'il détecte une identité dupliquée. Dans ce cas, il envoie l'identité id'_j chiffrée à CH_l afin d'effectuer une vérification

directe (cette étape permettant d'éviter les faux positifs induits par la probabilité de collisions existante dans un filtre de Bloom). Si CH_l confirme la collision d'identité, la dernière étape (étape (7)) est activée et la détection d'une attaque par réplication est signalée. Si la confirmation échoue, CH_l enregistre $id_r = id'_j$.

(7) Quand un nœud dupliqué est détecté dans le réseau, CH_l et CH_i (car le même mécanisme va être répété pour la vérification du filtre BF_i) démarrent ensemble un processus de révocation permettant d'isoler le nœud dupliqué id'_j.

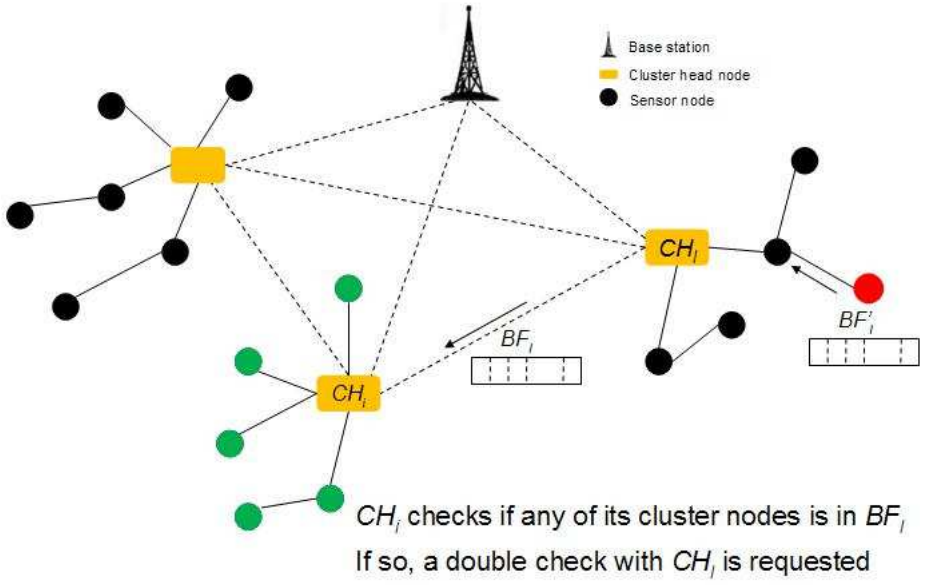


FIGURE 2.3 – Illustration de notre algorithme.

Je ne détaillerai pas ici l'analyse de sécurité qui permet d'affirmer que cette méthode fonctionne correctement, le lecteur peut se reporter à [ZMU09].

Évaluation théorique et simulations

Nous avons proposé deux évaluations de notre protocole : une évaluation théorique et une série de simulations.

La complexité générale de notre algorithme dépend essentiellement du nombre t de clusterheads. Chaque clusterhead envoie $2(t-1)$ messages. Le coût total en communication est donc de l'ordre de $\mathcal{O}(t^2)$ messages de taille m bits et la mémoire requise pour chaque clusterhead est $\mathcal{O}(t)$ messages de taille m bits. Ainsi, en termes de nombre de bits échangés et sans considérer le filtre de Bloom, notre algorithme est plus efficace que LSM dès que : $n\sqrt{n} \times |id| \geq t^2 \times \frac{n}{t}|id|$ où $|id|$ est la taille binaire d'une identité et où $d = \frac{n}{t}$ représente le nombre moyen de membres d'un cluster. Cela nous permet d'affirmer que notre algorithme est plus efficace que l'algorithme LSM dès que $t \leq \sqrt{n}$ (nous ne tenons pas compte ici de la complexité nécessaire à la construction des clusters eux-mêmes car nous supposons que notre algorithme ne sera utilisé que quand ce type de mécanisme est déjà mis en place dans le réseau).

En tenant compte du filtre de Bloom utilisé pour compresser l'information (il s'agit ici de déporter une partie de l'énergie consommée habituellement en transmission en calculs sur les nœuds), on obtient comme borne : $n\sqrt{n} \times |id| \geq t^2 \times \frac{-\frac{n}{t}\ln p}{(\ln 2)^2}$ car la taille optimale m d'un filtre

de Bloom, étant donné N le nombre d'éléments insérés et une probabilité p de faux positifs est $m = -\frac{N \ln p}{(\ln 2)^2}$. Dans ce cas, notre algorithme est plus efficace que LSM dès que $t \leq \frac{\sqrt{n}|id|(\ln 2)^2}{-\ln p}$ en considérant $k = 7$ applications de la fonction de hachage pour la construction du filtre. Cela signifie que notre algorithme restera plus efficace que LSM dès que de gros clusters sont construits, c'est-à-dire essentiellement pour des réseaux denses ou de grands réseaux.

Cependant, notre algorithme à l'étape 5 doit réaliser des opérations supplémentaires pour trouver une identité particulière appartenant au filtre de Bloom reçu. Le nombre de calculs de hachage à effectuer est donc $k \cdot \frac{n}{|S_l|}$.

Nous avons également réalisé un certain nombre de simulations sous WSNet afin de comparer notre proposition aux algorithmes RM et LSM présentés dans [PPG05], le mécanisme d'élection de clusterheads sous-jacent étant ici LNCA [XV07]. Ces résultats sont donnés dans la thèse de W. Znaidi [Zna10]. En résumé, notre algorithme étant déterministe contrairement à RM et LSM, il aura toujours un meilleur taux de détection de l'attaque par réplication même quand beaucoup de couples de nœuds sont répliqués (ou quand un même nœud est répliqué un grand nombre de fois) que ce soit dans un modèle de propagation idéal ou dans un modèle réaliste. De plus, même si dans notre algorithme, le nombre de bits par message est plus important, le nombre de messages envoyés est moins important que dans le cas de RM ou LSM et le gain en terme d'énergie totale dépensée (considérant chaque bit envoyé et chaque bit reçu dans le réseau) entre RM ou LSM et notre algorithme est supérieur à 1 dans tous les cas et augmente avec le nombre de nœuds considérés dans le réseau.

Ainsi, il semble que dès que l'on considère un réseau de capteurs hiérarchisé, notre solution est mieux adapté que les deux meilleures solutions existantes pour détecter des attaques par réplication de nœuds.

2.3 Résilience pour les protocoles de routage dans les réseaux de capteurs sans fil

Dans le cadre du projet ANR ARESA2 et de la thèse financée par Orange Labs Grenoble de Ochirkhand Erdene-Ochir (2009-2012) co-encadrée par Fabrice Valois (pour le CITI), Apostolos Kountouris (pour Orange Labs) et moi-même, nous nous intéressons à un but plus large. Nous cherchons à définir des conditions qui vont rendre les protocoles de routage utilisés classiquement dans les réseaux de capteurs sans fil plus résilients aux attaques. C'est-à-dire que nous cherchons des moyens de continuer à garantir le service "routage" même en présence d'attaquants dans le réseau. Plus précisément, nous considérons que les nœuds peuvent être compromis (mais pas la station de base) et que les attaquants sont du type *selective forwarding*, *sinkhole* ou *wormhole*. Il s'agit donc de proposer des protocoles de routage qui vont permettre de contourner correctement ces attaques afin de continuer à délivrer les données collectées au puits.

Dans les études réalisées jusqu'à présent [EOMVK10b, EOMVK10a], nous avons tout d'abord donné notre propre définition de la résilience ainsi qu'une métrique la représentant [EOMVK12] et ensuite, nous avons testé la résilience de protocoles de routage existants puis nous avons modifié ces protocoles afin de les rendre plus résilients si possible à moindre coût énergétique [EOMVK10c, EOMVK11].

2.3.1 Définition de la résilience et proposition de métrique

En mécanique, la résilience est la propriété d'un matériau à retrouver sa taille et sa forme après une déformation causée par une compression. Nous avons défini, par analogie, la résilience

comme la capacité d'un réseau de capteurs à "continuer à fournir le service demandé" (dans notre cas, il s'agira du routage) en présence de k attaquants. En d'autres termes, il s'agit de la capacité d'un réseau de capteurs à supporter des attaques internes et à continuer à délivrer les paquets des sources au puits en présence de k nœuds compromis.

Notons qu'habituellement dans la littérature, la résilience [Wag04, GGSE01, LY06] ne s'applique pas au cas du routage mais concerne plutôt l'agrégation de données, le routage tolérant aux fautes ou les schémas de prédistribution de clés. Notons cependant que dans [Wag04], D. Wagner compare la résilience de plusieurs fonctions d'agrégation. Nous souhaiterions comparer, durant notre étude, la résilience de plusieurs variantes d'un protocole de routage via une métrique unique.

Nous avons donc défini notre métrique de la résilience comme une surface à plusieurs axes où chacun des axes va représenter un paramètre de performance du routage sur le réseau. Plus la surface sera grande, plus la résilience du protocole sera élevée et plus la différence de surfaces entre un protocole confronté à k attaquants et ce même protocole confronté à m attaquants avec $m > k$ sera petite, plus le protocole sera résilient. Plus précisément et afin que le calcul de surface soit indépendant de la position des axes sur le graphique, la surface est calculée comme la somme des surfaces de tous les triangles possibles pouvant être construits à partir des 5 axes.

Comme un routage résilient se définit par sa capacité à continuer à acheminer les données 1) avec succès 2) efficacement 3) équitablement pour tous les nœuds, nous avons choisi les performances suivantes, classiquement utilisées en réseau, comme axes de notre métrique :

- Le succès de livraison de paquets est représenté par le taux de livraison moyen défini comme le nombre de paquets reçus par le puits divisé par le nombre de paquets envoyés par toutes les sources.
- L'efficacité dépend de plusieurs paramètres. Nous avons retenu ceux qui nous semblaient les plus pertinents pour un réseau de capteurs, à savoir l'efficacité énergétique, le délai et le débit moyen.
- L'équité vue ici comme le fait que toutes les sources peuvent atteindre le puits dépend de ce que nous avons nommé l'équité de livraison définie comme l'écart type du taux de livraison de toutes les sources à la moyenne.

Nous illustrerons la pertinence de cette métrique à l'aide des exemples définis par la suite.

2.3.2 Les protocoles de routage étudiés et leurs versions résilientes

Dans [EOMVK10b, EOMVK10a], nous avons étudié les protocoles de routage classiques Dynamic Source Routing (DSR) [JM96], Gradient-Based Routing (GBR) [MS01], Greedy forwarding (Greedy) [KK00] et Random Walk Routing (RWR) [SB02]. Les trois premiers fonctionnent sur le principe du plus court chemin, seul le moyen de construire ce chemin diffère :

- Dans DSR, un nœud souhaitant communiquer avec le puits inonde le réseau avec un `ROUTE REQUEST`, lorsque le puits reçoit cette demande, il choisit la route la plus courte et la renvoie via un `ROUTE REPLY`, c'est ensuite cette route qui sera utilisée pour acheminer les paquets.
- Dans GBR, c'est le puits qui initie un calcul de gradient à partir de lui-même (en commençant à 0) via un paquet `INTEREST` qui enregistre le nombre de sauts depuis le puits. Chaque nœud recevant ce paquet enregistre sa propre hauteur puis transmet ce paquet en ayant augmenté de 1 la valeur du gradient contenu dans le paquet `INTEREST` et ainsi de suite créant ainsi un plus court chemin au puits.
- Dans Greedy forwarding, les données permettant la construction du plus court chemin sont les localisations géographiques des nœuds et du puits.
- Quant à Random Walk, il utilise un principe bien différent : un nœud souhaitant envoyer

une donnée au puits choisit aléatoirement un de ses voisins pour transmettre cette donnée et ainsi de suite.

Nous avons étudié ces 4 protocoles de routage au regard de notre métrique, en considérant k (k représentant entre 10 et 50% des nœuds du réseau) attaquants de type *selective forwarding* présents dans le réseau. Des simulations obtenues à l'aide du simulateur WSNet, nous avons pu déduire que les trois premiers protocoles utilisant la métrique du plus court chemin avaient des résultats extrêmement similaires tandis que les résultats obtenus pour RWR étaient moins bons en raison de l'allongement des routes qui augmente la probabilité de croiser un attaquant. Cependant, le choix déterministe d'une route est une mauvaise propriété de résilience car si un attaquant est présent sur cette route, tous les paquets entre le puits et un nœud particulier seront perdus. La redondance structurelle de la topologie physique ne sera pas exploitée. RWR est quant à lui capable d'exploiter cette redondance mais souffre de l'allongement des routes. De même, la réplication de paquets permettra via la diversification de chemins proposés d'atteindre plus souvent le puits, même si dans ce cas, il faut tenir compte de la dépense énergétique.

Dans [EOMVK10c, EOMVK11], nous nous sommes donc attachés à modifier ces protocoles afin d'augmenter leurs capacités de résilience en tenant compte des remarques précédentes. Pour cela, nous proposons que le choix du nœud suivant sur la route soit fait de manière aléatoire dans l'ensemble des voisins qui sont plus proches du puits. De plus, nous avons également modifié les protocoles probabilistes précédents en ajoutant (ou non) de la réplication de paquets. Cette réplication est fonction de la distance du nœud au puits (ainsi un nœud à d sauts du puits enverra $d-1$ copies de son paquet).

Nous avons simulé l'ensemble des scénarios possibles sous WSNet [HCG08] en considérant entre 10 et 50 % d'attaquants pour un réseau de 300 nœuds déployés sur un périmètre de taille $100m \times 100m$ où chaque nœud a une portée radio de $20m$ et un degré moyen de 31. Les résultats sont présentés aux Figures 2.4, 2.5, 2.6, 2.7 et 2.8. On peut, par exemple, observer sur ces figures que, dans tous les cas, la surface est plus grande quand il n'y a pas d'attaque et que les variantes randomisées avec réplication sont celles qui garantissent la plus grande surface quand 30% d'attaquants sont présents dans le réseau. On observe également un certain nombre d'effets de bord. Par exemple, la valeur du coefficient DE (efficacité du délai) augmente avec le nombre d'attaquants dans le réseau (cette variation est très notable pour RWR). Cela est lié au fait, que si beaucoup d'attaquants sont présents dans le réseau, beaucoup de paquets vont se perdre et les paquets qui vont arriver sont ceux envoyés par les voisins les plus proches du puits qui ont moins de chance de trouver un attaquant sur leur route. Ainsi, comme ce coefficient ne tient compte que des paquets qui arrivent, ceux qui y parviennent, le font plus rapidement. Le même genre de paradigme reste vrai dans le cas de l'efficacité énergétique (EE) où plus il va y avoir d'attaquants dans le réseau, plus le nombre de paquets perdus augmente et donc moins le réseau va consommer d'énergie. Une méthode simple pour empêcher ces effets de bord est de diviser la valeur obtenue par le nombre de paquets arrivés au puits pour les deux axes.

En utilisant les calculs de surfaces présentés aux Figures 2.7 et 2.8, la résilience, par analogie avec le test de Charpy [13], peut être vue comme le coefficient directeur de la droite reliant deux points entre eux dans les Figures 2.7 et 2.8 : plus le coefficient directeur est grand, meilleure est la résilience. On peut ainsi remarquer que les protocoles classiques sont généralement moins résilients que les versions randomisées de ces mêmes protocoles, elles-mêmes, moins résilientes que les versions randomisées avec réplication fonction de la distance au puits. On peut également remarquer qu'au sens de notre métrique, même sans attaquant, les versions seulement randomi-

13. voir par exemple : `http://fr.wikipedia.org/wiki/Essai_de_flexion_par_choc_sur_%C3%A9prouvette_entaill%C3%A9e_Charpy`

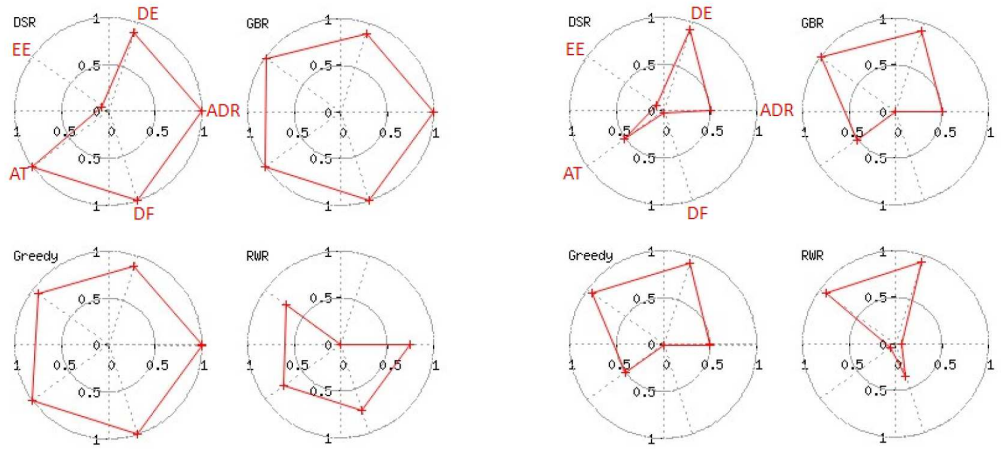


FIGURE 2.4 – Protocoles classiques : surface de résilience (a) sans attaque ($k = 0\%$) (b) avec attaques ($k = 30\%$). ADR : taux de livraison moyen, AT : débit moyen, DF : équité de livraison, EE : efficacité énergétique, DE : efficacité en terme de délai.

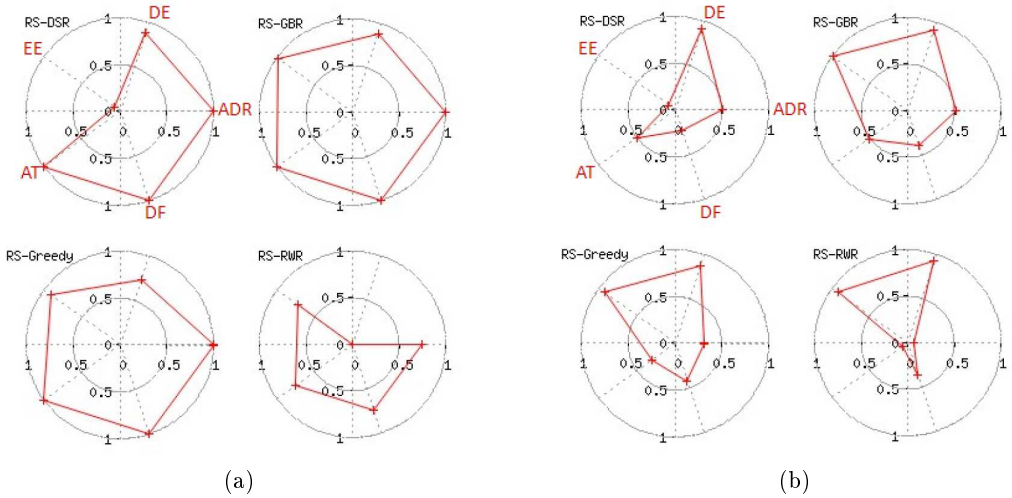
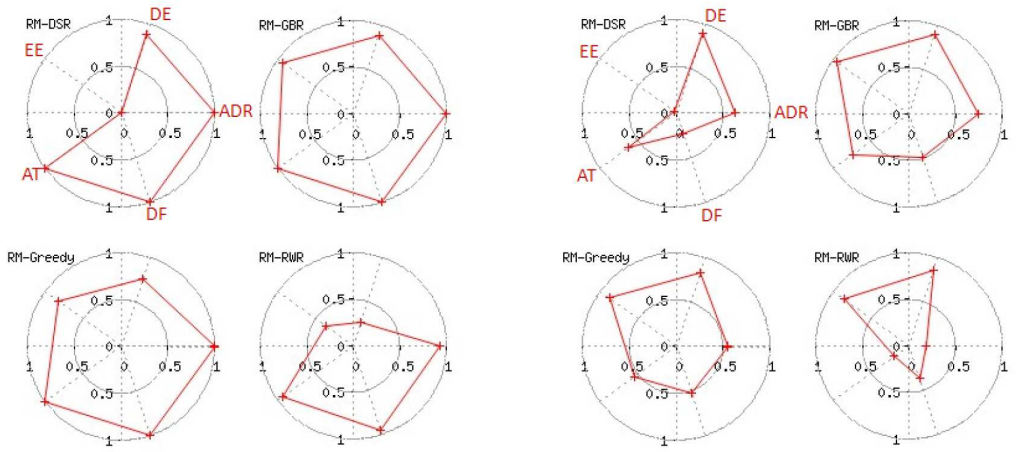


FIGURE 2.5 – Versions randomisées des protocoles classiques : surface de résilience (a) sans attaque ($k = 0\%$) (b) avec attaques ($k = 30\%$).

sées de DSR et GBR restent pertinentes pour faire du routage. Notre étude a également montré que tous les résultats décrits ici sont encore plus marqués lorsque l'on considère des attaques *sinkhole* ou *wormhole*.

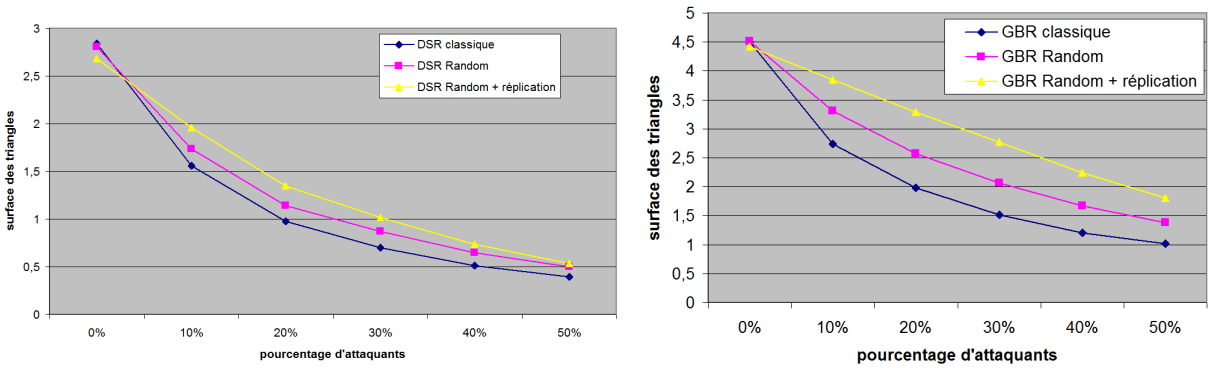
2.3.3 Travaux futurs

Il reste bien sûr à définir un cadre théorique pour valider ces travaux de simulation, en nous inspirant des travaux de D. Wagner dans [Wag04]. Pour cela, nous considérons que le réseau est un graphe et nous calculons les probabilités de croiser un attaquant choisi aléatoirement dans le graphe sur une route donnée. Nous souhaitons également mener plus avant les travaux concernant



(a) (b)

FIGURE 2.6 – Versions randomisées des protocoles classiques avec réplication fonction de la distance : surface de résilience (a) sans attaque ($k = 0\%$) (b) avec attaques ($k = 30\%$).



(a) (b)

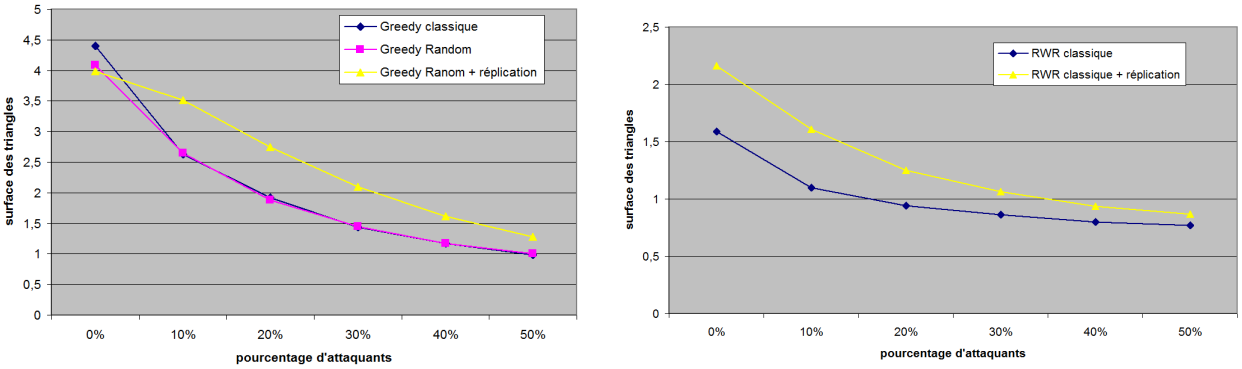
FIGURE 2.7 – surface de résilience en fonction du pourcentage d'attaquants pour les trois variantes de chaque protocole : (a) pour DSR et (b) pour GBR.

les méthodes de réplication afin d'essayer de déterminer lesquelles sont les plus optimales.

2.4 Problématiques de sécurité dans le codage réseau

Dans le cadre du projet ARESA2 et du post-doctorat de Yuan Yuan Zhang, nous avons mené plusieurs études sur la sécurité pour le codage réseau. Cette technique permet grâce à la dissémination d'information dans un réseau de capteurs (il s'agit de découper une donnée en plusieurs morceaux qui seront combinés/codés ensemble afin que ces derniers soient disséminés et routés dans le réseau en multicast orienté) permet d'améliorer le débit des transmissions d'information multicast ou unicast dans un réseau formé d'une ou de plusieurs sources et d'un ou de plusieurs récepteurs comme prouvé dans [MK02, KM03].

Évidemment, qui dit nouvelles techniques pour les réseaux dit nouvelles attaques. Ainsi les attaques par pollution ou les attaques par inondation décrites dans la suite de cette section sont



(a) (b)

FIGURE 2.8 – surface de résilience en fonction du pourcentage d'attaquants pour les trois variantes de chaque protocole : (a) pour Greedy et (b) pour RWR.

des attaques dédiées au codage réseau. Nous nous sommes donc particulièrement employés à développer des méthodes permettant de lutter contre ces nouvelles attaques. Nous les détaillerons dans cette section après avoir rappelé quelques définitions.

2.4.1 Définition du codage réseau

L'article fondateur concernant le codage réseau est celui de Ahlswede et al. [ACLY00]. Le but principal du codage réseau est de trouver une dissémination optimale de l'information dans un réseau. Il a été montré que le codage réseau (voir [HL08] Chapitre 1) peut également améliorer la résilience du réseau contre les pannes de communication ou les effacements. Les réseaux filaires ou sans fil peuvent bénéficier des avantages du codage réseau (voir par exemple [HL08, YLCZ05, CDFZ06]).

L'une des méthodes les plus importantes pour le codage réseau est l'utilisation de codes linéaires : les paquets échangés par les nœuds sont des combinaisons linéaires sur un corps fini de données à transmettre. Les codes linéaires aléatoires [HMK+06] ont particulièrement retenu l'attention. En considérant le réseau vu comme un graphe avec un nœud source et des nœuds destination, le codage s'effectue comme suit : Soit $D = (d_1, d_2, \cdots, d_n)$ une donnée de kn bits vue comme un vecteur de n fragments $d_i \in \mathbb{F}_{2^k},\ i \in [1, n]$. Les messages $m_j = h_j \parallel p_j$ transmis par la source et relayés par les nœuds intermédiaires en utilisant des codes linéaires aléatoires sont : une entête h_j et une donnée p_j avec $p_j = \sum_{i=1}^{n} \alpha_{i,j} d_i$, où les coefficients $\alpha_{i,j}$ sont tirés au hasard dans $\mathbb{F}_q$ avec $q = 2^u$. L'entête quant à elle est composée de tous les coefficients $\alpha_{i,j}$ permettant de décrire la donnée : $h_j = (\alpha_{1,j}, \cdots, \alpha_{n,j})$.

La source et les nœuds relais appliquent le processus de codage à l'infini jusqu'à ce qu'ils reçoivent de la destination un accusé de réception (ACK) de toutes les destinations. Ces dernières décodent les messages reçus (quand elles en ont reçu suffisamment qui sont linéairement indépendants) via une élimination gaussienne ou une autre méthode de résolution de systèmes linéaires.

Il y a deux types de flots qui peuvent être considérés : le codage réseau inter-flots qui combine des messages de différentes sources et le codage réseau intra-flots (aussi appelé codage réseau source) qui combine entre eux uniquement les messages d'un seul flot. Il existe beaucoup de résultats concernant le codage réseau et ses applications, notamment sur les bornes théoriques de débit atteignables que je ne détaillerai pas ici spécialisant mon travail sur la sécurité du codage

réseau. Pour plus de détails, le lecteur peut se reporter à [HL08] par exemple.

2.4.2 Attaques par pollution et attaques par inondation contre le codage réseau

Le codage réseau en lui-même permet de se prémunir contre un certain nombre d'attaques. On peut citer notamment ici l'attaquant "wiretapper" qui fait de l'écoute passive et qui tant qu'il n'a pas récupéré suffisamment de paquets linéairement indépendants ne peut pas accéder à l'information complète (pour plus de détails voir [OW85, CY11, ES08]). De même, en raison de la nature multicast de la méthode d'envoi des messages, un attaquant de type *selective forwarding* ne va avoir pour seul effet qu'une augmentation du délai avant que la ou les destinations décodent correctement la donnée (voir par exemple [DCNR09]).

Les attaques dédiées au codage réseau [DCNR09] sont essentiellement de deux types et sont des attaques actives. Il s'agit d'une part de l'attaque par pollution et d'autre part de l'attaque par inondation. Dans une attaque par pollution, un adversaire injecte des messages invalides (les paquets pollués) dans le flot de messages. Il exploite ainsi la capacité du codage réseau à disséminer l'information à son propre avantage. Les messages invalides vont être mélangés à des paquets valides en aval et vont rapidement polluer l'ensemble des paquets du réseau. En outre, le destinataire qui obtient plusieurs paquets n'a aucun moyen de distinguer lesquels parmi ceux-ci sont valides ou non et peuvent être utilisés ou non pour le décodage. En effet, même avec un seul paquet pollué utilisé durant le décodage, tous les messages vont être incorrectement décodés. Les ressources du réseau en termes d'énergie et de bande passante vont donc être gaspillées. Beaucoup de solutions ont été proposées dans la littérature pour se prémunir contre ce type d'attaques. On peut citer ici [YWRG08, YWRG09, BFKW09, CJL09, AB09] qui utilisent pour la plupart des signatures homomorphiques permettant de vérifier l'intégrité des messages. Dans [AZF+10] et dans sa version étendue [AZF+11], nous avons proposé une solution permettant de prévenir les attaques par pollution dans le cas du codage réseau par ou exclusif. Cette solution sera détaillée dans la section 2.4.4.

L'attaque par inondation [DCNR09] (que nous avions également nommée bien mal à propos attaque Tsunami) se focalise quant à elle sur les accusés de réception (paquets ACK en anglais) que la destination doit renvoyer à la source lorsqu'elle a correctement décodé la donnée. Les attaques par inondation sont très faciles à mettre en place dans les réseaux sans fil en raison de l'utilisation des communications radio (l'attaquant peut facilement écouter le paquet ACK). Pour lancer une attaque par inondation, un adversaire peut soit modifier le contenu du paquet ACK, soit injecter de faux paquets ACK, soit supprimer le ACK ou retarder sa livraison :

- *Injecter de faux paquets ACK :* dans ce cas, l'attaquant se contente de forger un faux paquet ACK et de l'envoyer à la source qui croira le message correctement décodé par la destination et passera au message suivant. Ainsi, la destination ne pourra pas décoder correctement le message. On peut facilement empêcher ce type d'attaque en authentifiant le paquet ACK via l'utilisation d'un code d'authentification de message [KBC97].
- *Supprimer des paquets ACK :* dans ce cas, l'adversaire détruit le paquet ACK qu'il doit transmettre ou modifie le chemin de livraison du paquet ACK afin d'empêcher qu'il ne parvienne jusqu'à la source. Ainsi, la source continuera indéfiniment à envoyer des paquets codés à la destination gaspillant par la même les ressources du réseau.
- *Retarder des paquets ACK :* dans ce cas, au lieu de supprimer le paquet ACK, l'attaquant retarde juste sa livraison. Cette conduite est difficile à distinguer d'une conduite égoïste où les nœuds veulent réduire leur propre consommation énergétique. Cette attaque augmentera le temps nécessaire à la transmission d'un message (augmentant les délais de bout en bout)

tout en gaspillant les ressources du réseau.

Dans [ZZLM11], nous avons proposé de transférer le paquet ACK en multi-chemin afin d'éviter les attaques par inondation. Ce résultat sera détaillé dans la section 2.4.5.

2.4.3 Solutions cryptographiques pour la sécurité du codage réseau

Avant de rentrer à proprement parler dans les solutions de sécurité pour le codage réseau que nous avons développées, cette partie présente les solutions cryptographiques les plus récentes, intéressantes pour une utilisation dans le codage réseau. En effet, afin d'être applicables sur des codes linéaires sur $\mathbb{F}_2$ ou sur des corps plus gros, les chiffrements et les méthodes d'authentification considérés doivent vérifier certaines propriétés d'homomorphisme. Nous détaillons donc ici quelques solutions existantes.

Chiffrement homomorphique

Les données dans le codage réseau peuvent avoir besoin d'être cependant chiffrées par exemple lorsqu'on considère un attaquant extrêmement puissant, même si le codage réseau permet une confidentialité intrinsèque. Dans ce cas, il est nécessaire d'utiliser pour garantir une confidentialité de bout en bout des propriétés d'homomorphisme et d'homomorphisme complet.

Un schéma de chiffrement E, comme défini dans [FG07], d'un ensemble M de textes clairs vers un ensemble C de textes chiffrés est dit homomorphique si pour toute clé de chiffrement donnée k, E satisfait :

$$\forall m_1, m_2 \in M, \quad E(m_1 \odot_M m_2) \leftarrow E(m_1) \odot_C E(m_2)$$

pour des opérateurs $\odot_M$ de M et $\odot_C$ de C où $\leftarrow$ signifie "peut directement être calculé à partir de" (i.e. sans déchiffrement intermédiaire).

Cette définition reste très générale et la plupart du temps, les schémas de chiffrement homomorphique sont utiles si les opérateurs $\odot_M$ et $\odot_C$ sont des opérateurs de groupe ou de corps tels que $+$ ou $\times$. On peut donc raffiner la définition précédente de deux manières :

- Un schéma de chiffrement est dit additivement homomorphique si $\forall m_1, m_2 \in M$, $E(m_1 +_M m_2) \leftarrow E(m_1) +_C E(m_2)$.
- Un schéma de chiffrement est dit multiplicativement homomorphique si $\forall m_1, m_2 \in M$, $E(m_1 \times_M m_2) \leftarrow E(m_1) \times_C E(m_2)$.

Beaucoup de schémas de chiffrement sont soit homomorphiquement additif soit homomorphiquement multiplicatif. Pour une étude complète sur les schémas homomorphiques, le lecteur peut se référer à [FG07]. Parmi les schémas homomorphiques à clé publique les plus connus, on peut citer le schéma de Goldwasser-Micali [GM82] qui utilise un nombre RSA et des calculs de symboles de Jacobi et le schéma de Paillier [Pai99] qui est additivement homomorphique. Il existe également un chiffrement additivement homomorphique simple : il s'agit du chiffrement à masque jetable détaillé au chapitre 1. Ce dernier convient très bien lorsque le codage réseau se fait sur $\mathbb{F}_2$ muni de l'opérateur $\oplus$ et même lorsque le codage réseau est inter-flots car le chiffrement à masque jetable vérifie également la propriété suivante : $\forall m_1, m_2 \in M, E_{k_1 \oplus k_2}(m_1 \oplus m_2) \leftarrow E_{k_1}(m_1) \oplus E_{k_2}(m_2)$, ce qui n'est pas le cas pour les autres schémas qui ne peuvent être utilisés qu'en intra-flots.

Plus récemment, en 2009, dans [Gen09], C. Gentry a introduit une notion plus puissante que celle définie précédemment qui s'appelle schéma de chiffrement complètement homomorphique. Ce schéma permet à quelqu'un d'évaluer des circuits sur des données chiffrées sans faire appel au déchiffrement. En d'autres termes, un schéma de chiffrement complètement homomorphique permet à toute personne de transformer publiquement une collection de textes chiffrés de textes

clairs $\pi_1, \cdots, \pi_n$ en un texte chiffré pour une fonction/un circuit $f(\pi_1, \cdots, \pi_n)$ des textes clairs, sans que la personne en question n'est besoin de la connaissance des textes clairs. Cette notion est bien sûr plus forte que la "simple" propriété homomorphique précédemment définie. Dans [Gen09], partant de cette définition, l'auteur présente un schéma complètement homomorphique construit à partir de réseaux Euclidiens idéaux. Malheureusement, ce schéma ne peut pas être utilisé en pratique car il a vraiment une très grande complexité en terme de calculs. Depuis ce travail fondateur, d'autres propositions ont été faites, essentiellement fondées sur les réseaux Euclidiens [SV10, DGHV10, SS10]. Ce domaine de recherche est réellement nouveau et est donc amené à se développer. Cependant, à l'heure actuelle, les schémas de chiffrement complètement homomorphiques proposés restent théoriques et très coûteux.

Authentification homomorphique

Afin de se prémunir contre les attaques par pollution d'un attaquant externe, le réseau doit être capable de garantir une authentification (ou une intégrité) saut à saut afin que l'attaque puisse être confinée à une partie du réseau. Pour cela, il est nécessaire d'introduire les nouveaux outils cryptographiques que sont les MAC homomorphiques, les fonctions de hachage homomorphiques et les signatures homomorphiques. Nous introduisons ici ces différentes notions. Rappelons cependant que dans le cas de l'utilisation de MAC une clé secrète unique doit être partagée entre tous les vérifieurs ou alors il est nécessaire de construire de nombreuses paires de clés.

Définition 2.4.1 (MAC Homomorphiques [LG10]) *Un MAC homomorphique doit satisfaire les propriétés suivantes :*

1. ***Homomorphisme.*** *Etant donné deux paires (message,tag)* $(\mathbf{m}_1, t_1)$ *et* $(\mathbf{m}_2, t_2)$*, n'importe qui peut créer un tag valide* t_a *pour un message agrégé* $\mathbf{m}_a = \omega_1\mathbf{m}_1 + \omega_2\mathbf{m}_2$ *pour tout scalaire* ω_1 *et* ω_2*. Typiquement,* $t_a = \omega_1 t_1 + \omega_2 t_2$*.*
2. ***Sécurité contre les Attaques à Messages Choisis.*** *Même sous une attaque à messages choisis, dans laquelle l'adversaire est autorisé à demander un nombre de tags polynomial en le nombre de messages, il est toujours impossible pour cet adversaire de créer un tag valide pour un message autre qu'une combinaison linéaire de messages précédemment demandés.*

Un MAC homomorphique consiste en la donnée de trois algorithmes probabilistes et en temps polynomial ($\mathsf{Sign}, \mathsf{Combine}, \mathsf{Verify}$*) :*

- $t_u = \mathsf{Sign}(k, \mathsf{rid}, \mathbf{m}_u, \mathsf{id}_u)$ *: le nœud u avec l'ID* id_u*, contributeur du message* $\mathbf{m}_u$ *concernant le rapport* rid*, calcule un tag* t_u *pour* $\mathbf{m}_u$ *en utilisant k comme clé.*
- $t = \mathsf{Combine}((\mathbf{m}_1, t_1, \omega_1), \cdots, (\mathbf{m}_n, t_n, \omega_n))$ *: un combineur implémente la propriété homomorphique pour la paire (message,tag) en l'absence de la clé k, autrement dit, il génère le tag t pour un message combiné* $\mathbf{m} = \sum_{i=1}^{n} \omega_i \mathbf{m}_i$*.*
- $\mathsf{Verify}(k, \mathsf{rid}, \mathbf{m}, t)$ *: un vérifieur vérifie l'intégrité du message* $\mathbf{m}$ *au vue du rapport* rid *à l'aide de la clé k et du tag t.*

Cette définition est une version modifiée de celle de [LG10] dont la première version a été publiée dans [AB09]. Dans [AB09], les auteurs proposent des exemples de tels schémas en se fondant sur les fonctions de hachage universelles proposées à l'origine par Carter et Wegman dans [CW79]. Ces constructions utilisent un générateur pseudo-aléatoire et une fonction pseudo-aléatoire. Un autre schéma basé sur deux générateurs pseudo-aléatoires a également été proposé dans [LG10].

Historiquement, les deux premiers schémas proposés dans la littérature étaient de simples MAC additivement homomorphiques sur $\mathbb{F}_2$ (dans ce cas les coefficients ω_i de la définition sont

0 ou 1 et l'opérateur + est un XOR). Ils ont été proposés par Krawczyk en 1994 [Kra94] en utilisant les définitions de fonctions de hachage universelles introduites par Carter et Wegman dans [CW77, CW79] et des codes de contrôles (CRC) à polynômes secrets ou des matrice de Toeplitz à initialisations secrètes. Ces schémas peuvent être directement utilisés pour empêcher les attaques par pollution dans le codage réseau utilisant le XOR comme proposé dans notre article [AZF+10] et détaillé dans la section 2.4.4.

Définition 2.4.2 (Fonctions de Hachage Homomorphiques [LG10, DGHV10, KFM04]) *Une fonction de hachage homomorphique H est une fonction de hachage qui vérifie :*

1. ***Homomorphisme.*** *Pour toute paire de messages $\mathbf{m}_1$, $\mathbf{m}_2$ et toute paire de scalaires ω_1, ω_2, on doit avoir $H(\omega_1\mathbf{m}_1+\omega_2\mathbf{m}_2) = H(\mathbf{m}_1)^{\omega_1}H(\mathbf{m}_2)^{\omega_2}$ (Notons que l'opération de groupe considérée ici est la multiplication).*
2. ***Résistance aux Collisions.*** *Il n'existe pas d'adversaire probabiliste en temps polynomial (PPT) capable de forger $(\mathbf{m}_1, \mathbf{m}_2, \mathbf{m}_3, \omega_1, \omega_2)$ satisfaisant à la fois $\mathbf{m}_3 \neq \omega_1\mathbf{m}_1 + \omega_2\mathbf{m}_2$ et $H(\mathbf{m}_3) = H(\mathbf{m}_1)^{\omega_1}H(\mathbf{m}_2)^{\omega_2}$.*

Des fonctions de hachage homomorphiques vérifiant la définition 2.4.2 peuvent être utilisées dans de nombreux domaines comme la sécurisation du codage réseau contre les attaques par pollution comme proposé dans [DGHV10] mais aussi la sécurisation des contenus pair-à-pair en utilisant des codes à effacement [KFM04]. D'autres exemples de fonctions de hachage homomorphiques sont donnés dans [GR05, GR06b] mais les deux schémas proposés sont lents. Dans [LG10], les auteurs proposent deux autres fonctions de hachage homomorphiques. Les méthodes utilisées pour construire de telles fonctions sont le logarithme discret et la factorisation d'entiers.

Définition 2.4.3 (Schémas de Signature pour le codage réseau [BFKW09, JMSW02]) *Un schéma de signature pour le codage réseau est un quadruplet d'algorithmes probabilistes en temps polynomial (*Setup, Sign, $\mathsf{Combine}$, Verify*) avec avec les fonctionnalités suivantes :*

- $\mathsf{Setup}(1^k, N)$. *Cet algorithme prend en entrée un paramètre de sécurité 1^k, un entier N et fournit en sortie un nombre premier p, une clé publique PK et une clé secrète SK.*
- $\mathsf{Sign}(SK, \mathsf{id}, \mathbf{m})$. *Cet algorithme prend en entrées une clé secrète SK, un identifiant de fichier $\mathsf{id} \in \{0,1\}^k$, et une vecteur $\mathbf{m} \in \mathbb{F}_p^N$ et fournit en sortie une signature σ.*
- $\mathsf{Combine}(PK, \mathsf{id}, \{(\omega_i, \sigma_i)\}_{i=1}^n)$. *Cet algorithme prend en entrées une clé publique PK, un identifiant de fichier $\mathsf{id} \in \{0,1\}^k$, et un ensemble $\{\omega_i, \sigma_i\}_{i=1}^n$ avec $\omega_i \in \mathbb{F}_p$. Il fournit en sortie une signature σ. (L'intuition est que si chaque σ_i est une signature valide d'un vecteur $\mathbf{m}_i$, alors σ est une signature valide de $\sum_{i=1}^n \omega_i\mathbf{m}_i$.)*
- $\mathsf{Verify}(PK, \mathsf{id}, \mathbf{y}, \sigma)$. *Cet algorithme prend en entrées une clé publique PK, un identifiant $\mathsf{id} \in \{0,1\}^k$, un vecteur $\mathbf{y} \in \mathbb{F}_p^N$, et une signature σ. Il fournit en sortie un booléen 0 (rejeter) ou 1 (accepter).*

Il est requis que pour chaque triplet (p, PK, SK) de sortie de $\mathsf{Setup}(1^k, N)$, les propriétés suivantes soient vérifiées :

- *Pour tout id et tout $\mathbf{y} \in \mathbb{F}_p^N$, si $\sigma \leftarrow \mathsf{Sign}(SK, \mathsf{id}, \mathbf{y})$ alors $\mathsf{Verify}(PK, \mathsf{id}, \mathbf{y}, \sigma) = 1$.*
- *Pour tout $\mathsf{id} \in \{0,1\}^k$ et tout triplet $\{(\omega_i, \sigma_i, \mathbf{m}_i)\}_{i=1}^n$, si l'égalité suivante est vérifiée $\mathsf{Verify}(PK, \mathsf{id}, \mathbf{m}_i, \sigma_i) = 1$ pour tout i, alors*

$$\mathsf{Verify}(PK, \mathsf{id}, \sum_{i=1}^{n} \omega_i\mathbf{m}_i, \mathsf{Combine}(PK, \mathsf{id}, \{(\omega_i, \sigma_i)\}_{i=1}^n)) = 1$$

Dans [BFKW09], les auteurs proposent deux constructions qui vérifient la définition précédente pour un unique nœud source. Ces constructions utilisent des couplages bilinéaires sur des courbes elliptiques. Le premier schéma proposé NCS_1 a l'avantage que les signatures produites peuvent être associées à des vecteurs individuels plus qu'à un sous-espace entier. Dans [LG10], les auteurs montrent que la construction proposée dans [GR06a] qui est un schéma de signature fondé sur l'identité vérifie la définition précédente. Dans [ABBF10] et dans [BF11], les auteurs étendent les propositions de signatures pour le codage réseau faites dans [BFKW09] aux cas de codages réseaux multi-sources et multi-fichiers à travers une construction générique, la deuxième proposition utilisant des réseaux Euclidiens sur des corps binaires.

2.4.4 Authentification pour l'agrégation de données et le codage réseau

Même si les méthodes d'agrégation dans les réseaux de capteurs et le codage réseau sont deux concepts très différents en terme de choix topologiques et du graphe optimal sous-jacent, d'un point de vue sécurité, on peut comparer les propriétés requises par ces deux types de technique. C'est pourquoi, nous présentons dans une même section les résultats publiés dans [ZLM09] concernant l'authentification pour l'agrégation de données et ceux obtenus dans [AZF+10, AZF+11] concernant l'authentification pour le codage réseau XOR. Nous présentons également une méthode de lutte contre l'attaque par inondation.

Authentification pour l'agrégation de données

Les réseaux de capteurs sans fil sont souvent de grande taille et comme leur but est de collecter et de transmettre des données physiques vers un ou plusieurs points de collecte, la quantité d'information circulant dans le réseau peut être importante. De plus, les nœuds ont des capacités de calculs et d'énergie limitées. Il s'agit donc de trouver des mécanismes permettant de diminuer et de faciliter la manipulation de grandes quantités d'information en multi-saut afin d'augmenter la durée de vie du réseau. L'agrégation de données est une méthode classique permettant cela. Elle consiste, pour chaque nœud parent qui reçoit n messages $(x_1, x_2, \cdots, x_n)$ de ses n fils, à agréger ces messages à l'aide d'une fonction f (qui peut être la somme, la moyenne, etc.) afin de ne transmettre qu'un seul message m : $m = f(x_1, x_2, \cdots, x_n)$ et ainsi de suite jusqu'au puits. Dans ce contexte, la sécurité représente un problème critique. Ainsi, la confidentialité, l'intégrité et l'origine des données agrégées doivent être préservées et vérifiées. Pour cela, il s'agit d'utiliser d'une part les chiffrements homomorphiques présentés à la section 2.4.3 et également les MAC homomorphiques présentés à la même section.

L'utilisation du chiffrement à flot pour garantir la confidentialité de bout en bout et saut à saut pour l'agrégation des données dans un réseau de capteurs a été proposée dans [CMT05] et dans [Cas07, CCMT09]. On considère ici un nœud agrégateur i recevant un message m_{i-1}. Le nœud i agrège sa contribution x_i au message m_{i-1} de la manière suivante : $m_i = m_{i-1} + c_i \mod p = m_{i-1} + x_i + keystream_i \mod p$ où p est un nombre premier bien choisi commun à tous les nœuds et où $keystream_i$ est une valeur aléatoire générée par le nœud i à l'aide de sa clé secrète k_i et d'un algorithme de chiffrement à flot. Quant à m_{i-1} il s'agit bien évidemment d'un paquet précédent agrégé en utilisant le même processus.

Dans [ZLM09], avec W. Znaidi et C. Lauradoux, nous avons proposé d'utiliser les constructions MAC décrites dans [Kra94] pour garantir l'authentification de l'agrégation de valeurs xorées. Nous avons également généralisé à $\mathbb{F}_p$ (p premier) les constructions de [Kra94] afin de pouvoir considérer une agrégation sur $\mathbb{F}_p$ et combiner la méthode de chiffrement présentée ci-dessus avec une authentification garantie par les MAC homomorphiques. On considère, pour

montrer comment fonctionne cette technique, un réseau extrêmement simple composé de trois nœuds i, j et k, le nœud agrégateur. On suppose également, que les messages (par exemple des relevés de température) vont être envoyés par paquets, chaque paquet étant constitué de l messages de taille p. Ainsi, chaque capteur relevant par exemple l températures stocke ces l valeurs $M = (M_0, \cdots, M_{l-1})$ avant de les envoyer toutes ensemble à la station de base à chaque intervalle de temps donné ou à la demande. Chaque nœud aura également en commun avec la station de base une clé de chiffrement K_E, une valeur commune d'initialisation IV qui doit absolument être changée à chaque chiffrement, une clé d'authentification K_A, un même algorithme de chiffrement à flot E et enfin un polynôme irréductible commun à tous les nœuds et à la station de base $q(x)$ de degré n à coefficients dans $\mathbb{F}_p$ permettant d'utiliser les CRC cryptographiques proposés dans [Kra94].

Le nœud i après avoir stocké l messages les envoie à la station de base en les chiffrant et en les authentifiant. Pour le chiffrement, il emploie la méthode d'agrégation de chiffrement proposée dans [CMT05] c'est-à-dire que pour chaque message M_j pour j variant de 0 à $l-1$, il calcule $M_j + r_j \mod p$ où r_j est une suite pseudo-aléatoire produite par $E(K_{Ei}, IV_i)$ et comprise entre 0 et $p-1$ où p est un nombre premier bien choisi. Le nœud transmet également à la station de base son identité i et la valeur unique IV_i, un compteur. Le nœud i transmet donc à son père k, $C^i = M^i + r^i = (C^i_0 = M^i_0 + r^i_0 \mod p, \cdots, C^i_{l-1} = M^i_{l-1} + r^i_{l-1} \mod p)$. Le nœud i calcule ensuite le MAC associé à l'ensemble de ces messages : tag^i. Le nœud i transmet finalement à son nœud parent k : $\{hdr, data, tag\}$ avec $hdr = i||IV_i$, $data = C^i$, $tag = tag^i$. Le nœud j fait de même et envoie : : $\{j||IV_j, C^j, tag^j\}$. Le nœud k transmet alors à la station de base (en considérant que lui-même doit envoyer l messages M^k) : $\{i||IV_i, j||IV_j, k||IV_k, C^i + C^j + C^k, tag^i + tag^j + tag^k\}$. La station de base déchiffre alors $C^i + C^j + C^k - r^i - r^j - r^k = M^i + M^j + M^k = M$ à l'aide de sa connaissance des différentes clés de chiffrement et des différentes valeurs d'IV. Elle obtient $M = (\sum_i M^i_0 \mod p, \cdots, \sum_i M^i_{l-1} \mod p)$ et elle vérifie ensuite les tags :

$$\begin{aligned} tag^i + tag^j + tag^k &= M^i \cdot x^n + M^j \cdot x^n + M^k \cdot x^n \mod q(x) \\ &= (M^i + M^j + M^k) \cdot x^n \mod q(x) \\ &= (\sum_i M^i_0 \mod p, \cdots \sum_i M^i_{l-1} \mod p) \cdot x^n \mod q(x) \\ &= M \cdot x^n \mod q(x) \end{aligned}$$

Ainsi, la station de base peut vérifier la valeur des tags agrégés en fonction de la valeur de la somme reçue.

Dans [ZLM09], nous proposons plusieurs évaluations de cette méthode tant théorique que pratique. Clairement, la méthode présentée ici ne permettra pas de garantir la propriété de non-malléabilité (i.e. un attaquant ne pourra pas modifier des données mais pourra ajouter ou supprimer des données valides). Cependant cette méthode reste intéressante car elle prévient la modification des données. En ce qui concerne les résultats de simulations, nous comparons des méthodes utilisant de l'agrégation pour le chiffrement et/ou pour l'authentification. Sans surprise, ce sont les méthodes qui intégrent à la fois de l'agrégation pour les données chiffrées et l'authentification qui sont les plus efficaces. Nous montrons également dans [AZF+10, AZF+11] que les performances logicielles du MAC proposé par Krawczyk et fondé sur des matrices de Toeplitz sont meilleures que celle de HMAC [KBC97] utilisant SHA-1 [oC95] et de CBC-MAC utilisant l'AES [FIP01] (voir également annexe B).

Authentification pour le codage réseau

Dans [AZF+10, AZF+11] présenté dans sa version conférence en annexe B, avec A. Apavatjrut, W. Znaidi, A. Fraboulet, C. Goursaud, C. Lauradoux et K. Jaffrès-Runser, nous avons étudié les différentes stratégies possibles d'authentification permettant de se prémunir contre les attaques par pollution dans les réseaux de capteurs quand le codage réseau XOR est utilisé. Nous définissons donc tout d'abord les 4 stratégies d'authentification possibles (avec des MAC homomorphiques ou non) illustrées sur le modèle de base (le réseau linéaire) du codage réseau. Dans ce modèle Alice et Bob envoient respectivement x_1 et x_2 à Eve, le nœud relais, qui *broadcaste* ensuite $x_1 \oplus x_2$ aux deux nœuds. Notons que pour que toutes ces méthodes fonctionnent, Alice, Bob et Eve doivent partager une clé secrète k, commune à tous les trois.

Les quatre stratégies sont :

- **Authenticate-Xor-MAC-Forward (AXMF)** : Dans ce mode illustré à la Fig. 2.9, Alice et Bob envoient à Eve leurs deux messages x_1 et x_2 ainsi que les MAC correspondants. Eve vérifie d'abord les MAC de chacun des messages reçus puis calcule le xor de x_1 et x_2, puis le MAC de $x_1 \oplus x_2$ et *broadcaste* ces deux données. Alice et Bob n'ont alors plus qu'à vérifier le MAC reçu puis à extraire la valeur manquante. Ce mode fonctionne avec n'importe quel MAC (HMAC et CBC-MAC par exemple) qu'il soit homomorphique ou non.
- **Authenticate-Xor-Forward (AXF)** : Dans ce mode illustré à la Fig. 2.9, Alice et Bob envoient à Eve leurs deux messages x_1 et x_2 ainsi que les MAC correspondants. Eve vérifie d'abord les MAC de chacun des messages reçus puis calcule le xor de x_1 et x_2, puis le XOR des MAC de x_1 et x_2 et *broadcaste* ces deux données. Alice et Bob n'ont alors plus qu'à vérifier le MAC reçu. Ce mode fonctionne uniquement avec un MAC homomorphique.
- **Xor-Authenticate-Forward (XAF)** : Dans ce mode illustré à la Fig. 2.10, Alice et Bob envoient à Eve leurs deux messages x_1 et x_2 ainsi que les MAC correspondants. Eve calcule tout d'abord le XOR de x_1 et x_2, puis le XOR des MAC reçus. Elle vérifie ensuite si ce XOR de MAC est valide ou non. Si c'est le cas, elle renvoie ces deux valeurs à Alice et Bob qui n'ont plus qu'à vérifier le MAC reçu. Ce mode fonctionne uniquement avec un MAC homomorphique.
- **Xor-Forward (XF)** : Dans ce mode, Alice et Bob envoient à Eve leurs deux messages x_1 et x_2 ainsi que les MAC correspondants. Eve calcule tout d'abord le XOR de x_1 et x_2, puis le XOR des MAC reçus. Elle renvoie ces deux valeurs à Alice et Bob qui n'ont plus qu'à vérifier le MAC reçu. Ce mode fonctionne uniquement avec un MAC homomorphique. Ce mode reporte toutes les vérifications sur les nœuds destinataires car Eve n'authentifie pas les messages qu'elle reçoit. Les paquets pollués ne sont donc pas détectés. Cependant, ce mode est efficace en terme d'énergie et peut être combiné avec le mode XAF pour trouver un compromis entre le coût énergétique quand aucune attaque ne se produit et le coût en énergie quand un adversaire injecte des paquets corrompus.

Dans [AZF+10, AZF+11], nous avons ensuite comparé différents MAC (HMAC utilisant SHA-1, CBC-MAC utilisant AES et le MAC homomorphique de H. Krawczyk et fondé sur des matrices de Toeplitz) en terme de performances énergétique et également les différentes stratégies d'authentification possibles. Il s'est avéré que le mode le plus économique est bien évidemment XF qui ne fait aucune vérification sur le nœud relai suivi du mode XAF. Les modes XAF et AXF utilisant les matrices de Toeplitz permettent également de réduire de 42% à 68% la consommation d'énergie pour nœuds relais par rapport à XAF avec CBC-MAC ou HMAC. Pour plus de détails sur les simulations, le lecteur peut se reporter à l'annexe B. Nous en avons déduit que si la probabilité qu'une attaque par pollution se produise dans le réseau est inférieure à 31%,

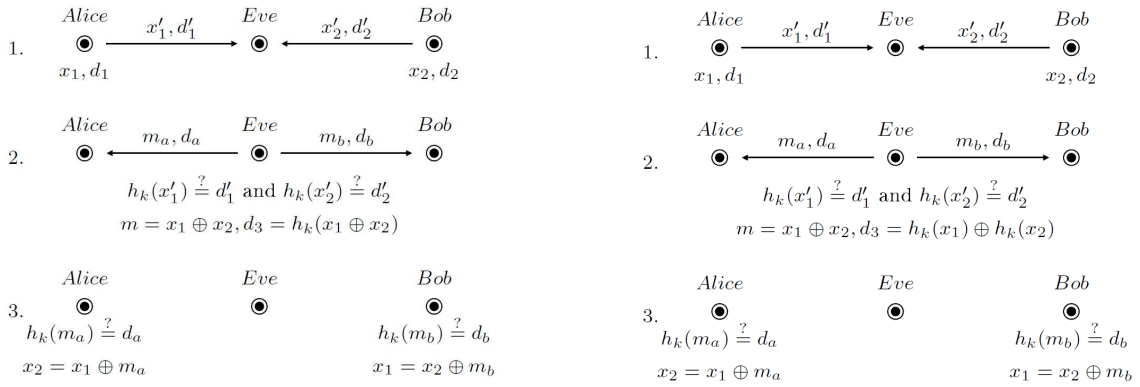


FIGURE 2.9 – Les deux premières stratégies : AUTHENTICATE-XOR-MAC-FORWARD à gauche et AUTHENTICATE-XOR-FORWARD à droite

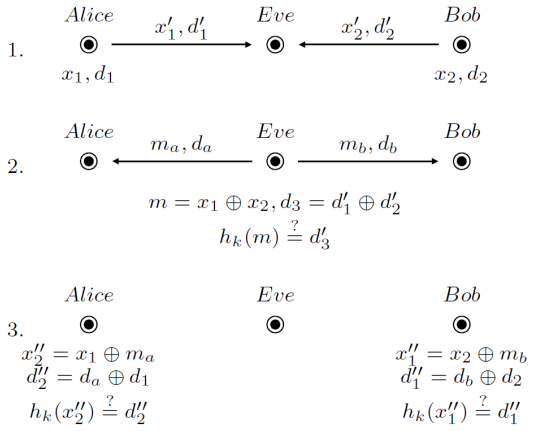


FIGURE 2.10 – La troisième stratégie : XOR-AUTHENTICATE-FORWARD.

il vaut mieux utiliser XF sinon, c'est XAF qui doit être privilégié. Dans la version étendue de cet article de conférence [AZF+11], nous avons inclus une étude permettant de limiter le nombre de vérifications de MAC à faire dans le mode XAF lorsque le MAC agrégé n'est pas valide. Ceci se fait à l'aide des méthodes de "group testing" qui ont un coût moyen linéaire. Nous avons également étendu l'étude concernant le choix des modes en fonction de la probabilité d'attaques au cas d'un réseau papillon.

2.4.5 Attaques par inondation dans le codage réseau

Dans [ZZLM11], avec Y. Zhang, W. Znaidi et Cédric Lauradoux, nous nous sommes intéressés à l'attaque par inondation produite lorsqu'un attaquant détruit ou retarde un paquet ACK de récépicé. Pour cela, nous avons modifié légèrement le principe de base du codage réseau en incluant dans l'entête des paquets codés le chemin qu'ils ont parcouru jusqu'au puits. Le puits enregistre alors toutes les routes prises par un message provenant d'une source particulière. Une

fois que le puits a réussi à correctement décoder le message envoyé par une source, il utilise 1, 2, 3, 4 ou 5 chemins de retour pour faire parvenir le ACK à la source et ainsi éviter une attaque par inondation.

Les résultats des simulations effectuées sous WSNet en considérant un modèle de propagation idéal et les couches physique et MAC de IEEE 802.11 montrent que le gain en terme de messages qui sont décodés avec succès et où la source reçoit bien le ACK est de l'ordre d'entre 10% et 25% pour une proportion d'attaquants entr 10% et 30% quand 1 seul paquet ACK ou 5 paquet ACK sont envoyés. Nous avons testé des réseaux de 50, 100, 150 et 200 nœuds et plus le réseau est dense, plus le gain est important. Cela provient du fait que pour des réseaux de petites tailles il n'est pas toujours évident de construire 5 chemins distincts entre une source et une destination.

Nous avons déjà commencé à étendre ce travail en regardant également l'influence d'attaquants de type *selective forwarding* sur le délai induit pour que le puits décode correctement le message. Nous souhaiterions à terme inclure une étude théorique permettant d'intégrer aux modèles classiquement utilisés dans le codage réseau des attaquants qui agissent avec une certaine probabilité p.

Bibliographie

[AB05a] François Arnault et Thierry P. Berger : Design and Properties of a New Pseudorandom Generator Based on a Filtered FCSR Automaton. *IEEE Transaction on Computers*, 54(11):1374–1383, 2005.

[AB05b] François Arnault et Thierry P. Berger : F-FCSR : design of a new class of stream ciphers. *In Fast Software Encryption - FSE 2005*, volume 3557 de *Lecture Notes in Computer Science*, pages 83–97. Springer, 2005.

[AB09] Shweta Agrawal et Dan Boneh : Homomorphic MACs : MAC-Based Integrity for Network Coding. *In Applied Cryptography and Network Security - ACNS 2009*, Lecture Notes in Computer Science 5536, pages 292–305. Springer, 2009.

[ABBF10] Shweta Agrawal, Dan Boneh, Xavier Boyen et David Mandell Freeman : Preventing pollution attacks in multi-source network coding. *In Public Key Cryptography - PKC 2010*, volume 6056 de *Lecture Notes in Computer Science*, pages 161–176. Springer, 2010.

[ABL06] François Arnault, Thierry P. Berger et Cédric Lauradoux : Update on F-FCSR Stream Cipher. ECRYPT - Network of Excellence in Cryptology, Call for stream Cipher Primitives - Phase 2 2006. `http://www.ecrypt.eu.org/stream/`.

[ABL+09] François Arnault, Thierry P. Berger, Cédric Lauradoux, Marine Minier et Benjamin Pousse : A new approach for FCSRs. *In Selected Areas in Cryptography - SAC 2009, Revised Selected Papers*, volume 5867 de *Lecture Notes in Computer Science*, pages 433–448, 2009.

[ABLM07] François Arnault, Thierry P. Berger, Cédric Lauradoux et Marine Minier : X-FCSR - a new software oriented stream cipher based upon FCSRs. *In Progress in Cryptology - INDOCRYPT 2007*, volume 4859 de *Lecture Notes in Computer Science*, pages 341–350, 2007.

[ABM08] François Arnault, Thierry P. Berger et Marine Minier : Some Results on FCSR Automata With Applications to the Security of FCSR-Based Pseudorandom Generators. *IEEE Transactions on Information Theory*, 54(2):836–840, 2008.

[ABMPar] François Arnault, Thierry P. Berger, Marine Minier et Benjamin Pousse : Revisiting LFSRs for cryptographic applications. *IEEE Transactions on Information Theory*, to appear. accepté en Juillet 2011.

[ABP11] François Arnault, Thierry P. Berger et Benjamin Pousse : A matrix approach for FCSR automata. *Cryptography and Communications*, 3(2):109–139, 2011.

[ACLY00] Rudolf Ahlswede, Ning Cai, Shuo-Yen Robert Li et Raymond W. Yeung : Network information flow. *IEEE Transactions on Information Theory*, 46(4): 1204–1216, 2000.

[AF03] G. Ars et J.-C. Faugère : An algebraic cryptanalysis of nonlinear filter generators using Gröbner bases. Research Report INRIA Lorraine, number 4739, 2003.

[AHMNP10] Jean-Philippe Aumasson, Luca Henzen, Willi Meier et María Naya-Plasencia : Quark : A lightweight hash. *In Cryptographic Hardware and Embedded Systems, CHES 2010*, volume 6225 de *Lecture Notes in Computer Science*, pages 1–15. Springer, 2010.

[AKK+10] Jean-Philippe Aumasson, Emilia Käsper, Lars R. Knudsen, Krystian Matusiewicz, Rune Steinsmo Ødegård, Thomas Peyrin et Martin Schläffer : Distinguishers for the Compression Function and Output Transformation of Hamsi-256. *In Information Security and Privacy - ACISP 2010*, volume 6168 de *Lecture Notes in Computer Science*. Springer, 2010.

[AZF+10] A. Apavatjrut, W. Znaidi, A. Fraboulet, C. Goursaud, C. Lauradoux et M. Minier : Energy friendly integrity for network coding in wireless sensor networks. *In International Conference on Network and System Security - NSS 2010*, pages 223–230. IEEE, September 2010.

[AZF+11] Apavatjrut Anya, Wassim Znaidi, Antoine Fraboulet, Claire Goursaud, Cédric Lauradoux, Marine Minier et Jaffrès-Runser Katia : Energy Efficient Authentication Strategies for Network Coding. *Concurrency and Computation : Practice and Experience*, Juin 2011. à paraître.

[BAK98] E. Biham, R. Anderson et L. Knudsen : SERPENT : A new block cipher proposal. *In Fast Software Encryption - FSE'98*, volume 1372 de *Lecture Notes in Computer Science*, pages 222–238. Springer, 1998.

[BBC+08a] C. Berbain, O. Billet, A. Canteaut, N. Courtois, H. Gilbert, L. Goubin, A. Gouget, L. Granboulan, C. Lauradoux, M. Minier, T. Pornin et H. Sibert : Sosemanuk, a fast software-oriented stream cipher. *In New Stream Cipher Designs - The eSTREAM Finalists*, volume 4986 de *Lecture Notes in Computer Science*, pages 98–118. Springer, 2008.

[BBC+08b] Côme Berbain, Olivier Billet, Anne Canteaut, Nicolas Courtois, Blandine Debraize, Henri Gilbert, Louis Goubin, Aline Gouget, Louis Granboulan, Cédric Lauradoux, Marine Minier, Thomas Pornin et Hervé Sibert : Decimv2. *In New Stream Cipher Designs - The eSTREAM Finalists*, volume 4986 de *Lecture Notes in Computer Science*, pages 140–151. Springer, 2008.

[BBD06] Alexander Becher, Zinaida Benenson et Maximillian Dornseif : Tampering with motes : Real-world physical attacks on wireless sensor networks. *In Security in Pervasive Computing, Third International Conference - SPC 2006*, volume 3934 de *Lecture Notes in Computer Science*, pages 104–118. Springer, 2006.

[BBG+09] Ryad Benadjila, Olivier Billet, Henri Gilbert, Gilles Macario-Rat, Thomas Peyrin, Matt Robshaw et Yannick Seurin : SHA-3 Proposal : ECHO. Submission to NIST (updated), 2009.

[BBS99a] Eli Biham, Alex Biryukov et Adi Shamir : Cryptanalysis of Skipjack Reduced to 31 Rounds Using Impossible Differentials. *In Advances in Cryptology - EUROCRYPT '99*, volume 1592 de *Lecture Notes in Computer Science*, pages 12–23. Springer, 1999.

[BBS99b] Eli Biham, Alex Biryukov et Adi Shamir : Miss in the Middle Attacks on IDEA and Khufu. *In Fast Software Encryption - FSE '99*, volume 1636 de *Lecture Notes in Computer Science*, pages 124–138. Springer, 1999.

[BCCM+08] Emmanuel Bresson, Anne Canteaut, Benoît Chevallier-Mames, Christophe Clavier, Thomas Fuhr, Aline Gouget, Thomas Icart, Jean-François Misarsky, Marìa Naya-Plasencia, Pascal Paillier, Thomas Pornin, Jean-René Reinhard, Céline Thuillet et Marion Videau : Shabal, a Submission to NIST's Cryptographic Hash Algorithm Competition. Submission to NIST, 2008.

[BCD03] Alex Biryukov, Christophe De Cannière et Gustaf Dellkrantz : Cryptanalysis of SAFER++. *In Advances in Cryptology - CRYPTO 2003*, volume 2729 de *Lecture Notes in Computer Science*, pages 195–211. Springer, 2003.

[BD07] Eli Biham et Orr Dunkelman : A Framework for Iterative Hash Functions - HAIFA. Cryptology ePrint Archive, Report 2007/278, 2007. `http://eprint.iacr.org/2007/278` (Accessed on 10/1/2010).

[BD09] Eli Biham et Orr Dunkelman : The SHAvite-3 Hash Function. Submission to NIST (Round 2), 2009. `http://www.cs.technion.ac.il/~orrd/SHAvite-3/Spec.15.09.09.pdf`.

[BDPA08] Guido Bertoni, Joan Daemen, Michael Peeters et Gilles Van Assche : On the indifferentiability of the sponge construction. *In Advances in Cryptology - EUROCRYPT 2008*, volume 4965 de *Lecture Notes in Computer Science*, pages 181–197. Springer, 2008.

[Ber08] Daniel J. Bernstein : The salsa20 family of stream ciphers. *In New Stream Cipher Designs - The eSTREAM Finalists*, volume 4986 de *Lecture Notes in Computer Science*, pages 84–97. Springer, 2008.

[BF01] Dan Boneh et Matthew K. Franklin : Identity-based encryption from the weil pairing. *In Advances in Cryptology - CRYPTO 2001*, volume 2139 de *Lecture Notes in Computer Science*, pages 213–229. Springer, 2001.

[BF11] Dan Boneh et David Mandell Freeman : Linearly homomorphic signatures over binary fields and new tools for lattice-based signatures. *In Public Key Cryptography - PKC 2011*, volume 6571 de *Lecture Notes in Computer Science*, pages 1–16. Springer, 2011.

[BFKW09] Dan Boneh, David Freeman, Jonathan Katz et Brent Waters : Signing a Linear Subspace : Signature Schemes for Network Coding. *In Public Key Cryptography - PKC 2009*, volume 5443 de *Lecture Notes in Computer Science*, pages 68–87. Springer, 2009.

[BG07] Côme Berbain et Henri Gilbert : On the Security of IV Dependent Stream Ciphers. *In Fast Software Encryption - FSE 2007*, volume 4593 de *Lecture Notes in Computer Science*, pages 254–273. Springer, 2007.

[Bih93] Eli Biham : New types of cryptoanalytic attacks using related keys (extended abstract). *In Advances in Cryptology - EUROCRYPT 1993*, volume 765 de *Lecture Notes in Computer Science*, pages 398–409. Springer, 1993.

[BK09] Alex Biryukov et Dmitry Khovratovich : Related-key cryptanalysis of the full AES-192 and AES-256. *In Advances in Cryptology - ASIACRYPT 2009*, volume 5912 de *Lecture Notes in Computer Science*, pages 1–18. Springer, 2009.

[BKL+07] Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin et C. Vikkelsoe : PRESENT : An Ultra-Lightweight Block Cipher. *In Cryptographic Hardware and Embedded Systems - CHES 2007*, Lecture Notes in Computer Science 4727, pages 450–466. Springer, 2007.

[BKM10] Chérifa Boucetta, Mohamed Ali Kaafar et Marine Minier : How secure are secure localization protocols in WSNs. *In International ICST Conference on Wireless Sensor Network (WSN) Systems and Software - S-Cube 2010*. ICST, December 2010.

[BKN09] Alex Biryukov, Dmitry Khovratovich et Ivica Nikolic : Distinguisher and related-key attack on the full AES-256. *In Advances in Cryptology - CRYPTO 2009*, volume 5677 de *Lecture Notes in Computer Science*, pages 231–249. Springer, 2009.

[BKR11] Andrey Bogdanov, Dmitry Khovratovich et Christian Rechberger : Biclique cryptanalysis of the full aes. Cryptology ePrint Archive, Report 2011/449, 2011. `http://eprint.iacr.org/`.

[BLM07] Chakib Bekara et Maryline Laurent-Maknavicius : A new resilient key management protocol for wireless sensor networks. *In Information Security Theory and Practices. Smart Cards, Mobile and Ubiquitous Computing Systems - WISTP 2007*, volume 4462 de *Lecture Notes in Computer Science*, pages 14–26. Springer, 2007.

[Blo70] Burton H. Bloom : Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426, 1970.

[Blu01] Bluetooth : Specification of the bluetooth system, volume 1 : Core, v1.1. Bluetooth SIG, February 2001.

[BLU07] Fatiha Benali, Véronique Legrand et Stéphane Ubéda : An ontology for the management of heteregenous alerts of information system. *In Proceedings of the 2007 International Conference on Security & Management - SAM 2007*, pages 374–380. CSREA Press, 2007.

[BM] Thierry P. Berger et Marine Minier : The GLUON family : a lightweight hash function family based on fcsrs. soumis.

[BM05] Thierry P. Berger et Marine Minier : Two algebraic attacks against the F-FCSRs using the IV mode. *In Progress in Cryptology - Indocrypt 2005*, volume 3797 de *Lecture Notes in Computer Science*, pages 143–154. Springer, 2005.

[BMP09] Thierry P. Berger, Marine Minier et Benjamin Pousse : Software Oriented Stream Ciphers Based upon FCSRs in Diversified Mode. *In Progress in Cryptology - INDOCRYPT 2009*, volume 5922 de *Lecture Notes in Computer Science*, pages 119–135. Springer, 2009.

[BMR04] Laurent Bussard, Refik Molva et Yves Roudier : History-based signature or how to trust anonymous documents. *In Second International Conference on Trust Management - iTrust 2004*, volume 2995 de *Lecture Notes in Computer Science*, pages 78–92. Springer, 2004.

[BR93] Mihir Bellare et Phillip Rogaway : Random Oracles are Practical : A Paradigm for Designing Efficient Protocols. *In ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[BS91] Eli Biham et Adi Shamir : Differential Cryptanalysis of DES-like Cryptosystems. *J. Cryptology*, 4(1):3–72, 1991.

[BS01] Alex Biryukov et Adi Shamir : Structural cryptanalysis of SASAS. *In Advances in Cryptology - EUROCRYPT 2001*, volume 2045 de *Lecture Notes in Computer Science*, pages 394–405. Springer, 2001.

[BSV+98] Carlo Blundo, Alfredo De Santis, Ugo Vaccaro, Amir Herzberg, Shay Kutten et Moti Yong : Perfectly secure key distribution for dynamic conferences. *Inf. Comput.*, 146(1):1–23, 1998.

[BTB04] Richard Beckwith, Dan Teibel et Pat Bowen : Report from the field : Results from an agricultural wireless sensor network. *In LCN '04 : Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks*, pages 471–478. IEEE Computer Society, 2004.

[BW99] Alex Biryukov et David Wagner : Slide attacks. *In Fast Software Encryption - FSE '99*, volume 1636 de *Lecture Notes in Computer Science*, pages 245–259. Springer, 1999.

[CA05] Yi Cheng et D.P. Agrawal : Efficient pairwise key establishment and management in static wireless sensor networks. *In Mobile Adhoc and Sensor Systems Conference, 2005. IEEE International Conference on*, pages 1–7. IEEE, 2005.

[Can06] Anne Canteaut : *Analyse et conception de chiffrements à clef secrète.* Hdr, Université Pierre et Marie Curie - Paris VI, 2006.

[Can07] C. De Cannière : Software performance of the phase 3 candidates. eSTREAM, ECRYPT Stream Cipher Project, 2007. `http://www.ecrypt.eu.org/stream/phase3perf.html`.

[Cap04] Licia Capra : Engineering human trust in mobile system collaborations. *In Proceedings of the 12th ACM SIGSOFT International Symposium on Foundations of Software Engineering - SIGSOFT FSE 2004*, pages 107–116. ACM, 2004.

[Cas07] Claude Castelluccia : Securing very dynamic groups and data aggregation in wireless sensor networks. *In The Fourth IEEE International Conference on Mobile Ad-hoc and Sensor Systems - MASS 2007*, pages 1–9. IEEE Computer Society, 2007.

[CBD+98] D. Coppersmith, C. Burwick, E. D'Avignon, R. Gennaro, S. Halevi, C. Jutla, S. Matyas Jr., L. O'Connor, M. Peyravian, D.Safford et N. Zunic : Mars - a Candidate Ciphe for AES. *In The First Advanced Encryption Standard Candidate Conference*. N.I.S.T., 1998.

[CCMT09] Claude Castelluccia, Aldar C-F. Chan, Einar Mykletun et Gene Tsudik : Efficient and provably secure aggregation of encrypted data in wireless sensor networks. *ACM Transactions on Sensor Networks*, 5(3):1–36, 2009.

[CDFZ06] Jillian Cannons, Randall Dougherty, Christopher F. Freiling et Kenneth Zeger : Network routing capacity. *IEEE Transactions on Information Theory*, 52(3):777–788, 2006.

[CDK09] Christophe De Cannière, Orr Dunkelman et Miroslav Knezevic : KATAN and KTANTAN - A Family of Small and Efficient Hardware-Oriented Block Ciphers. *In Cryptographic Hardware and Embedded Systems - CHES 2009*, Lecture Notes in Computer Science 5747, pages 272–288. Springer, 2009.

[CGH04] Ran Canetti, Oded Goldreich et Shai Halevi : On the random oracle methodology, revisited. *Journal of the ACM*, 51(4):557–594, 2004.

[CJL09] Denis Charles, Kamal Jain et Kristin Lauter : Signatures for network coding. *International Journal in Information and Coding Theory*, 1(1):3–14, 2009.

[CKM93] Don Coppersmith, Hugo Krawczyk et Yishay Mansour : The shrinking generator. *In Advances in Cryptology - CRYPTO 1993*, volume 773 de *Lecture Notes in Computer Science*, pages 22–39. Springer, 1993.

[CM03] N. Courtois et W. Meier : Algebraic attacks on stream ciphers with linear feedback. *In Advances in Cryptology - EUROCRYPT 2003*, volume 2656 de *Lecture Notes in Computer Science*, pages 345–359. Springer, 2003.

[CMT05] C. Castellucia, E. Mykletun et G. Tsudik : Efficient aggregation of encrypted data in wireless sensor networks. *In Mobile and Ubiquitous Systems : Networking and Services - MobiQuitous 2005*, pages 1–9. IEEE Computer Society, 2005.

[Cou03] N. Courtois : Fast algebraic attacks on stream ciphers with linear feedback. *In Advances in Cryptology - CRYPTO 2003*, volume 2729 de *Lecture Notes in Computer Science*, pages 177–194. Springer, 2003.

[CP08] Christophe De Cannière et Bart Preneel : Trivium. *In New Stream Cipher Designs - The eSTREAM Finalists*, volume 4986 de *Lecture Notes in Computer Science*, pages 244–266. Springer, 2008.

[CPS08] Jean-Sébastien Coron, Jacques Patarin et Yannick Seurin : The random oracle model and the ideal cipher model are equivalent. *In Advances in Cryptology - CRYPTO 2008*, volume 5157 de *Lecture Notes in Computer Science*, pages 1–20. Springer, 2008.

[CW77] Larry Carter et Mark N. Wegman : Universal Classes of Hash Functions (Extended Abstract). *In ACM Symposium on Theory of Computing - STOC '77*, pages 106–112. ACM, 1977.

[CW79] Larry Carter et Mark N. Wegman : Universal Classes of Hash Functions. *Journal of Computer and System Sciences - JCSS*, 18(2):143–154, 1979.

[CY11] Ning Cai et Raymond W. Yeung : Secure network coding on a wiretap network. *IEEE Transactions on Information Theory*, 57(1):424–435, 2011.

[CZK03] Xiofeng Chen, Fangguo Zhang et Kwandjo Kim : A new ID-based group signature scheme from bilinear pairings. *In Information Security Applications, 4th International Workshop - WISA'03*, volume 2908 de *Lecture Notes in Computer Science*, pages 585–592. Springer, 2003.

[Dam89] Ivan Damgård : A design principle for hash functions. *In Advances in Cryptology - CRYPTO '89*, volume 435 de *Lecture Notes in Computer Science*, pages 416–427. Springer, 1989.

[DCNR09] Jing Dong, Reza Curtmola et Cristina Nita-Rotaru : Secure network coding for wireless mesh networks : Threats, challenges, and directions. *Computer Communication*, 32(17):1790–1801, 2009.

[DGHV10] Marten Van Dijk, Craig Gentry, Shai Halevi et Vinod Vaikuntanathan : Fully homomorphic encryption over the integers. *In Advances in Cryptology - EUROCRYPT 2010*, volume 6110 de *Lecture Notes in Computer Science*, pages 24–43. Springer, 2010.

[DKR97] Joan Daemen, Lars R. Knudsen et Vincent Rijmen : The block cipher square. *In Fast Software Encryption - FSE'97*, volume 1267 de *Lecture Notes in Computer Science*, pages 149–165. Springer, 1997.

[DP10] W. Dargie et C. Poellabauer : *Fundamentals of Wireless Sensor Networks : Theory and Practice*. Wireless Communications and Mobile Computing. Wiley, 2010.

[DR98] Joan Daemen et Vincent Rijmen : AES proposal : Rijndael. *In The First Advanced Encryption Standard Candidate Conference*. N.I.S.T., 1998.

[DR02] Joan Daemen et Vincent Rijmen : *The Design of Rijndael : AES - The Advanced Encryption Standard*. Springer, 2002.

[Dub05] Dimitri Dubois : Confiance et population : doubles rôles et information dans le jeu de l'investissement répété. working paper, LAMETA, Université de Montpellier 1, Juin 2005.

[Dub06a] Dimitri Dubois : la confiance dans les interactions économiques et sociales. papier de recherche, LAMETA, Université de Montpellier 1, 2006.

[Dub06b] Dimitri Dubois : *L'émergence de relations de confiance-réciprocité au sein d'une population d'agents économiques*. Thèse de doctorat, Université Montpellier 1, UFR Sciences Economiques, 2006.

[EG02] L. Eschenauer et V.D. Gligor : A key-management scheme for distributed sensor networks. *In Proceedings of the 9th ACM Conference on Computer and Communications Security - CCS 2002*, pages 41–47. ACM, 2002.

[EJ02] P. Ekdahl et T. Johansson : A new version of the stream cipher SNOW. *In Selected Areas in Cryptography – SAC 2002*, volume 2295 de *Lecture Notes in Computer Science*, pages 47–61. Springer, 2002.

[EOMVK10a] Ochirkhand Erdene-Ochir, Marine Minier, Fabrice Valois et Apostolos Kountouris : Resiliency of Wireless Sensor Networks : Definitions and Analyses. *In IEEE 17th International Conference on Telecommunications - ICT 2010*, pages 828 – 835. IEEE, 2010.

[EOMVK10b] Ochirkhand Erdene-Ochir, Marine Minier, Fabrice Valois et Apostolos Kountouris : Resilient networking in wireless sensor networks. Rapport de recherche RR-7230, INRIA, 2010.

[EOMVK10c] Ochirkhand Erdene-Ochir, Marine Minier, Fabrice Valois et Apostolos Kountouris : Toward Resilient Routing in Wireless Sensor Networks : Gradient-Based Routing in Focus. *In Fourth International Conference on Sensor Technologies and Applications - SENSORCOMM 2010*, pages 478 – 483. IEEE, 2010.

[EOMVK11] Ochirkhand Erdene-Ochir, Marine Minier, Fabrice Valois et Apostolos Kountouris : Enhancing Resiliency Against Routing Layer Attacks inWireless Sensor Networks : Gradient-based Routing in Focus. *International Journal on Advances in Networks and Services*, 4(1 & 2):38 – 54, 2011. available online `http://www.iariajournals.org/networks_and_services/`.

[EOMVK12] Ochirkhand Erdene-Ochir, Marine Minier, Fabrice Valois et Apostolos Kountouris : A New Metric to Quantify Resiliency in Networking. *IEEE Communications Letters*, 2012. accepté en révision mineure.

[ES08] S.Y. El Rouayheb et E. Soljanin : On wiretap networks II. *In IEEE International Symposium on Information Theory, ISIT 2007*, pages 551–555. IEEE, 2008.

[eST08] ECRYPT Stream Cipher Project eSTREAM : The current estream portfolio. eSTREAM, ECRYPT Stream Cipher Project, 2008. `http://www.ecrypt.eu.org/stream`.

[FG07] Caroline Fontaine et Fabien Galand : A survey of homomorphic encryption for non-specialists. *EURASIP Journal on Information Security*, 1:1–15, 2007.

[FIP01] FIPS 197 : Advanced Encryption Standard. Federal Information Processing Standards Publication 197, 2001. U.S. Department of Commerce/N.I.S.T.

[FKL+01] Niels Ferguson, John Kelsey, Stefan Lucks, Bruce Schneier, Michael Stay, David Wagner et Doug Whiting : Improved cryptanalysis of rijndael. *In Fast Software Encryption - FSE 2000*, volume 1978 de *Lecture Notes in Computer Science*, pages 213–230. Springer, 2001.

[FMS08] Simon Fischer, Willi Meier et Dirk Stegemann : Equivalent Representations of the F-FCSR Keystream Generator. *In ECRYPT Network of Excellence - SASC Workshop*, pages 87–94, 2008. Available at `http://www.ecrypt.eu.org/stvl/sasc2008/`.

[FMU07] Nicolas Fournel, Marine Minier et Stéphane Ubéda : Survey and benchmark of stream ciphers for wireless sensor networks. *In Information Security Theory and Practices. Smart Cards, Mobile and Ubiquitous Computing Systems - WISTP 2007*, volume 4462 de *Lecture Notes in Computer Science*, pages 202–214. Springer, 2007.

[Fou99] The Free Software Foundation : The gnu privacy handbook. `http://www.gnupg.org/gph/en/manual.html`, 1999.

[Fro98] R. Froberg : *An Introduction to Gröbner Bases*. John Wiley & Sons, 8 1998.

[Fuh10] Thomas Fuhr : Finding Second Preimages of Short Messages for Hamsi-256. *In Advances in Cryptology - ASIACRYPT 2010*, volume 6477 de *Lecture Notes in Computer Science*, pages 20–37. Springer, 2010.

[Gal07] Samuel Galice : *Modèle dynamique de sécurité pour réseaux spontanés*. Thèse de doctorat, INSA de Lyon, 2007.

[Gam00] Diego Gambetta : Can we trust trust ? *In* Diego Gambetta, éditeur : *Trust : Making and Breaking Cooperative Relatioins*, chapitre 13, pages 213–237. Published Online, 2000.

[GB07] T. Good et M. Benaissa : Hardware results for selected stream cipher candidates. eSTREAM, ECRYPT Stream Cipher Project, SASC 2007, Report 2007/023, 2007. `http://www.ecrypt.eu.org/stream`.

[Gen03] Craig Gentry : Certificate-based encryption and the certificate revocation problem. *In Advances in Cryptology - EUROCRYPT 2003*, volume 2656 de *Lecture Notes in Computer Science*, pages 272–293. Springer, 2003.

[Gen09] Craig Gentry : Fully homomorphic encryption using ideal lattices. *In Proceedings of the 41st Annual ACM Symposium on Theory of Computing - STOC 2009*, pages 169–178. ACM, 2009.

[GG04] Solomon W. Golomb et Guang Gong : *Signal Design for Good Correlation : For Wireless Communication, Cryptography, and Radar*. Cambridge University Press, New York, NY, USA, 2004.

[GGSE01] Deepak Ganesan, Ramesh Govindan, Scott Shenker et Deborah Estrin : Highly-resilient, energy-efficient multipath routing in wireless sensor networks. *In Proceedings of the 2nd ACM Interational Symposium on Mobile Ad Hoc Networking and Computing - MobiHoc 2001*, pages 251–254. ACM, 2001.

[GK02] Mark Goresky et Andrew Klapper : Fibonacci and Galois representations of feedback-with-carry shift registers. *IEEE Transactions on Information Theory*, 48(11):2826–2836, 2002.

[GK09] M. Goresky et A. Klapper : Algebraic shift register sequences, 2009. Avalaible at `http://cs.engr.uky.edu/~klapper/algebraic.html`.

[GKM+08] Praveen Gauravaram, Lars R. Knudsen, Krystian Matusiewicz, Florian Mendel, Christian Rechberger, Martin Schläffer et Søren S. Thomsen : Grøstl – a SHA-3 candidate. Submission to NIST, 2008.

[GKM+11] Praveen Gauravaram, Lars R. Knudsen, Krystian Matusiewicz, Florian Mendel, Christian Rechberger, Martin Schläffer et Søren S. Thomsen : Grøstl – a SHA-3 candidate. Submission to NIST (Round 3), 2011.

[GLM+06] Samuel Galice, Véronique Legrand, Marine Minier, John Mullins et Stéphane Ubéda : A history-based framework to build trust management systems. *In Second International IEEE SECURECOMM Workshop on the Value of Security through Collaboration (SECOVAL 2006)*, august 2006.

[GLM+10] Praveen Gauravaram, Gaëtan Leurent, Florian Mendel, María Naya-Plasencia, Thomas Peyrin, Christian Rechberger et Martin Schläffer : Cryptanalysis of the 10-Round Hash and Full Compression Function of SHAvite-3-512. *In Progress in Cryptology - AFRICACRYPT 2010*, volume 6055 de *Lecture Notes in Computer Science*, pages 419–436. Springer, 2010.

[GLNU05] Samuel Galice, Véronique Legrand, Philippe Neuville et Stéphane Ubéda : Identification dans les réseaux spontanés. *In Conférence sur la Sécurité et Architectures Réseaux*, June 2005.

[GM82] Shafi Goldwasser et Silvio Micali : Probabilistic encryption and how to play mental poker keeping secret all partial information. *In Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing - STOC 1982*. ACM, 1982.

[GM08] Samuel Galice et Marine Minier : Improving integral attacks against rijndael-256 up to 9 rounds. *In Progress in Cryptology - AFRICACRYPT 2008*, volume 5023 de *Lecture Notes in Computer Science*, pages 1–15. Springer, 2008.

[GM09] Gérald Gavin et Marine Minier : Oblivious multi-variate polynomial evaluation. *In Progress in Cryptology - INDOCRYPT 2009*, volume 5922 de *Lecture Notes in Computer Science*, pages 430–442. Springer, 2009.

[GMMU06] Samuel Galice, Marine Minier, John Mullins et Stéphane Ubéda : Cryptographic protocol to establish trusted history of interactions. *In Workshop on Security and Privacy in Ad hoc and Sensor Networks - ESAS 2006*, volume 4357 de *Lecture Notes in Computer Science*, pages 136–149. Springer, 2006.

[GMU07] Samuel Galice, Marine Minier et Stéphane Ubéda : A trust protocol for community collaboration. *In IFIPTM - Trust Management*, volume 236 de *IFIP*, pages 169–184. Springer, 2007.

[Gol81] S. W. Golomb : *Shift Register Sequences*. Aegen Park Press, 1981.

[GP10] Henri Gilbert et Thomas Peyrin : Super-Sbox Cryptanalysis : Improved Attacks for AES-Like Permutations. *In Fast Software Encryption - FSE 2010*, volume 6147 de *Lecture Notes in Computer Science*, pages 365–383. Springer, 2010.

[GPP11] Jian Guo, Thomas Peyrin et Axel Poschmann : The PHOTON Family of Lightweight Hash Functions. *In Advances in Cryptology - CRYTPO 2011*, volume 6841 de *Lecture Notes in Computer Science*, pages 222–239. Springer, 2011.

[GR05] Christos Gkantsidis et Pablo Rodriguez : Network coding for large scale content distribution. *In 4th Annual Joint Conference of the IEEE Computer and Communications Societies - INFOCOM 2005*, pages 2235–2245. IEEE, 2005.

[GR06a] Craig Gentry et Zulfikar Ramzan : Identity-based aggregate signatures. *In Public Key Cryptography - PKC 2006*, volume 3958 de *Lecture Notes in Computer Science*, pages 257–273. Springer, 2006.

[GR06b] Christos Gkantsidis et Pablo Rodriguez : Cooperative security for network coding file distribution. *In 25th IEEE International Conference on Computer Communications - INFOCOM 2006*. IEEE, 2006.

[GS00] T. Grandison et M. Sloman : A survey of trust in internet application. *IEEE Communications Surveys & Tutorials*, 3(4), 2000.

[GSB+05] Aline Gouget, Hervé Sibert, Côme Berbain, Nicolas Courtois, Blandine Debraize et Chris J. Mitchell : Analysis of the bit-search generator and sequence compression techniques. *In Fast Software Encryption - FSE 2005*, volume 3557 de *Lecture Notes in Computer Science*, pages 196–214. Springer, 2005.

[Gün89] Christoph G. Günther : Parallel Generation of Recurring Sequences. *In Advances in Cryptology - EUROCRYPT'89*, volume 434 de *Lecture Notes in Computer Science*, pages 503–522, 1989.

[HBC01] Jean-Pierre Hubaux, Levente Buttyán et Srdjan Capkun : The quest for security in mobile ad hoc networks. *In Proceedings of the 2nd ACM Interational Symposium on Mobile Ad Hoc Networking and Computing - MobiHoc 2001*, pages 146–155. ACM, 2001.

[HCG08] E. Ben Hamida, G. Chelius et J.-M. Gorce : Scalability versus accuracy in physical layer modeling for wireless network simulations. *In 22nd ACM/IEEE/SCS Workshop on Principles of Advanced and Distributed Simulation (PADS 2008)*, 2008.

[HJ08] Martin Hell et Thomas Johansson : Breaking the F-FCSR-H Stream Cipher in Real Time. *In Advances in Cryptology - ASIACRYPT 2008*, volume 5350 de *Lecture Notes in Computer Science*, pages 557–569, 2008.

[HJ11] Martin Hell et Thomas Johansson : Breaking the Stream Ciphers F-FCSR-H and F-FCSR-16 in Real Time. *J. Cryptology*, 24(3):427–445, 2011.

[HJMM08] Martin Hell, Thomas Johansson, Alexander Maximov et Willi Meier : The grain family of stream ciphers. *In New Stream Cipher Designs - The eSTREAM Finalists*, volume 4986 de *Lecture Notes in Computer Science*, pages 179–190. Springer, 2008.

[HL08] Tracey Ho et Desmond Lun : *Network Coding : An Introduction*. Cambridge University Press, 2008.

[HLC+06] Pan Hui, J. Leguay, J. Crowcroft, J. Scott, T. Friedmani et V. Conan : Osmosis in pocket switched networks. *In Communications and Networking in China, 2006. ChinaCom '06*, pages 1–6, October 2006.

[HMK+06] Tracey Ho, Muriel Médard, Ralf Koetter, David R. Karger, Michelle Effros, Jun Shi et Ben Leong : A Random Linear Network Coding Approach to Multicast. *IEEE Transactions on Information Theory*, 52(10):4413–4430, 2006.

[HPJ04] Yih-Chun Hu, Adrian Perrig et David B. Johnson. : Packet leashes : A defense against wormhole attacks in wireless ad hoc networks. *In Proceedings of the Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), IEEE 2003, San Francisco, CA*, volume 3, pages 1976–1986. IEEE, 2004.

[HPJ05] Yih-Chun Hu, Adrian Perrig et David B. Johnson : Ariadne : a secure on-demand routing protocol for ad hoc networks. *Wirel. Netw.*, 11(1-2):21–38, 2005.

[HS05] Jin Hong et Palash Sarkar : New applications of time memory data tradeoffs. *In Advances in Cryptology - ASIACRYPT 2005*, volume 3788 de *Lecture Notes in Computer Science*, pages 353–372. Springer, 2005.

[HSH+06] Deukjo Hong, Jaechul Sung, Seokhie Hong, Jongin Lim, Sangjin Lee, Bonseok Koo, Changhoon Lee, Donghoon Chang, Jaesang Lee, Kitae Jeong, Hyun Kim, Jongsung Kim et Seongtaek Chee : HIGHT : A New Block Cipher Suitable for Low-Resource Device. *In Cryptographic Hardware and Embedded Systems - CHES 2006*, Lecture Notes in Computer Science 4249, pages 46–59. Springer, 2006.

[HV08] Karel Heurtefeux et Fabrice Valois : Distributed qualitative localization for wireless sensor networks. *In Ad-hoc, Mobile and Wireless Networks, 7th International Conference - ADHOC-NOW 2008*, volume 5198 de *Lecture Notes in Computer Science*, pages 218–229. Springer, 2008.

[Ind08] Sebastiaan Indesteege : The LANE hash function. Submission to NIST, 2008.

[JHK06a] Cees J.A. Jansen, Tor Helleseth et Alexander Kholosha : Cascade jump controlled sequence generator and pomaranch stream cipher (version 2). eSTREAM, ECRYPT Stream Cipher Project, Report 2006/006, 2006. `http://www.ecrypt.eu.org/stream`.

[JHK06b] Cees J.A. Jansen, Tor Helleseth et Alexander Kholosha : Pomaranch version 3. eSTREAM, ECRYPT Stream Cipher Project, 2006. `http://www.ecrypt.eu.org/stream`.

[JM96] D. B. Johnson et D. A. Maltz : Dynamic source routing in ad hoc wireless networks. *In Mobile Computing*, volume 353, pages 153–181. Springer US, 1996.

[JM05] E. Jaulmes et F. Muller : Cryptanalysis of ecrypt candidates F-FCSR-8 and F-FCSR-H. ECRYPT Stream Cipher Project Report 2005/046, 2005. `http://www.ecrypt.eu.org/stream`.

[JM06] E. Jaulmes et F. Muller : Cryptanalysis of the F-FSCR stream cipher family. *In Selected Areas in Cryptography - SAC 2005*, volume 3897 de *Lecture Notes in Computer Science*, pages 20–35. Springer, 2006.

[JMSW02] Robert Johnson, David Molnar, Dawn Xiaodong Song et David Wagner : Homomorphic signature schemes. *In Topics in Cryptology - CT-RSA 2002*, volume 2271 de *Lecture Notes in Computer Science*, pages 244–262. Springer, 2002.

[Jun03] Pascal Junod : On the optimality of linear, differential, and sequential distinguishers. *In Advances in Cryptology - EUROCRYPT 2003*, volume 2656 de *Lecture Notes in Computer Science*, pages 17–32. Springer, 2003.

[KBC97] Hugo Krawczyk, Mihir Bellare et Ran Canetti : HMAC : Keyed-Hashing for Message Authentication, 1997. RFC 2104.

[KFM04] Maxwell N. Krohn, Michael J. Freedman et David Mazières : On-the-fly verification of rateless erasure codes for efficient content distribution. *In IEEE Symposium on Security and Privacy - S&P 2004*, pages 226–240. IEEE Computer Society, 2004.

[KG93] Andrew Klapper et Mark Goresky : 2-adic shift registers. *In Fast Software Encryption - FSE'93*, volume 809 de *Lecture Notes in Computer Science*, pages 174–178. Springer, 1993.

[KG97] Andrew Klapper et Mark Goresky : Feedback shift registers, 2-adic span and combiners with memory. *Journal of Cryptology*, 10(2):111–147, 1997.

[KK00] Brad Karp et H. T. Kung : GPSR : greedy perimeter stateless routing for wireless networks. *In Proceedings of the 6th annual international conference on Mobile computing and networking - MOBICOM 2000*, pages 243–254, 2000.

[KKS01] John Kelsey, Tadayoshi Kohno et Bruce Schneier : Amplified boomerang attacks against reduced-round mars and serpent. *In Fast Software Encryption - FSE 2000*, volume 1978 de *Lecture Notes in Computer Science*, pages 75–93. Springer, 2001.

[KM03] Ralf Koetter et Muriel Médard : An algebraic approach to network coding. *IEEE/ACM Trans. Netw.*, 11(5):782–795, 2003.

[Knu95] Lars R. Knudsen : Truncated and higher order differentials. *In Fast Software Encryption - FSE 1994*, volume 1008 de *Lecture Notes in Computer Science*, pages 196–211. Springer, 1995.

[Knu99] Lars R. Knudsen : Contemporary block ciphers. *In Lectures on Data Security, Modern Cryptology in Theory and Practice, Summer School, Aarhus, Denmark, July 1998*, volume 1561 de *Lecture Notes in Computer Science*, pages 105–126. Springer, 1999.

[Kob97] Neal Koblitz : *p-adic numbers, p-adic analysis and Zeta-Functions*. Springer, 1997.

[KR07] Lars R. Knudsen et Vincent Rijmen : Known-key distinguishers for some block ciphers. *In Advances in Cryptology - ASIACRYPT 2007*, volume 4833 de *Lecture Notes in Computer Science*, pages 315–324. Springer, 2007.

[Kra94] Hugo Krawczyk : LFSR-based Hashing and Authentication. *In Advances in Cryptology - CRYPTO '94*, Lecture Notes in Computer Science 839, pages 129–139. Springer, 1994.

[Küc09] Özgül Kücük : The Hash Function Hamsi. Submission to NIST (updated), 2009.

[KW02] Lars R. Knudsen et David Wagner : Integral cryptanalysis. *In Fast Software Encryption - FSE 2002*, volume 2365 de *Lecture Notes in Computer Science*, pages 112–127. Springer, 2002.

[KW03] Chris Karlof et David Wagner : Secure routing in wireless sensor networks : Attacks and countermeasures. *Elsevier's AdHoc Networks Journal, Special Issue on Sensor Network Applications and Protocols*, 1(2–3):293–315, September 2003.

[LG10] Zhijun Li et Guang Gong : Data aggregation integrity based on homomorphic primitives in sensor networks. *In Ad-Hoc, Mobile and Wireless Networks, 9th International Conference - ADHOC-NOW 2010*, volume 6288 de *Lecture Notes in Computer Science*, pages 149–162. Springer, 2010.

[LH94] Susan K. Langford et Martin E. Hellman : Differential-linear cryptanalysis. *In Advances in Cryptology - CRYPTO '94*, volume 839 de *Lecture Notes in Computer Science*, pages 17–25. Springer, 1994.

[LHU03] Véronique Legrand, Dana Hooshmand et Stéphane Ubéda : Trusted ambient community for self-securing hybrid networks. Research Report 5027, INRIA, 2003.

[LK05] Chae Hoon Lim et Tymur Korkishko : mCrypton - A Lightweight Block Cipher for Security of Low-Cost RFID Tags and Sensors. *In Workshop on Information Security Applications - WISA 2005*, Lecture Notes in Computer Science 3786, pages 243–258. Springer Verlqg, 2005.

[LN03] Donggang Liu et Peng Ning : Establishing pairwise keys in distributed sensor networks. *In Proceedings of the 10th ACM conference on Computer and communications security - CCS 2003*, pages 52–61. ACM, 2003.

[LNAU03] Véronique Legrand, Farid Nait-Abdesselam et Stéphane Ubéda : Etablissement de la confiance et réseaux adhoc - un état de l'art. *In Conférence sur la Sécurité et Architectures Réseaux*, June 2003.

[LP73] T. G. Lewis et W. H. Payne : Generalized feedback shift register pseudorandom number algorithm. *J. ACM*, 20(3):456–468, 1973.

[LP11] Zhiqiang Lin et Dingyi Pei : Constructing a Diversified FCSR with a Given Connection Integer. Cryptology ePrint Archive, Report 2011/358, 2011. `http://eprint.iacr.org/`.

[LR88] M. Luby et C. Rackoff : How to construct pseudorandom permutations from pseudorandom functions. *SIAM Journal on Computing*, 17(2):373–386, 1988.

[LR08] Cédric Lauradoux et Andrea Röck : Parallel generation of ℓ-sequences. *In Sequences and Their Applications - SETA 2008*, volume 5203 de *Lecture Notes in Computer Science*, pages 299–312. Springer, 2008.

[LvHD+05] Yee Wei Law, Lodewijk van Hoesel, Jeroen Doumen, Pieter Hartel et Paul Havinga : Energy-efficient link-layer jamming attacks against wireless sensor network mac protocols. *In 3rd ACM workshop on Security of ad hoc and sensor networks - SASN 2005*, pages 76–88. ACM, 2005.

[LY06] X. Li et D. Yang : A quantitative survivability evaluation model for wireless sensor networks. *In IEEE International Conference on Networking, Sensing and Control*, pages 727–732, 2006.

[Mar03] G. Marsaglia : Xorshift RNGs. *Journal of Statistical Software*, 8(14):1–6, 2003.

[Mas93] J. Massey : SAFER K-64 : a Byte-oriented Block-ciphering Algorithm. *In Fast Software Encryption - FSE'93*, volume 809 de *Lectures Notes in Computer Science*, pages 1–17. Springer, 1993.

[Mat93] Mitsuru MATSUI : Linear cryptoanalysis method for des cipher. *In Advances in Cryptology - EUROCRYPT 1993*, volume 765 de *Lecture Notes in Computer Science*, pages 386–397. Springer, 1993.

[Mat94] Mitsuru MATSUI : The first experimental cryptanalysis of the data encryption standard. *In Advances in Cryptology - CRYPTO '94*, volume 839 de *Lecture Notes in Computer Science*, pages 1–11. Springer, 1994.

[Mat97] M. MATSUI : New Block Encryption Algorithm MISTY. *In Fast Software Encryption - FSE'97*, volume 1267 de *Lectures Notes in Computer Science*, pages 54–68. Springer, 1997.

[MB05] Jochen MUNDINGER et Jean-Yves Le BOUDEC : Analysis of a Reputation System for Mobile Ad-Hoc Networks with Liars. *In 3rd International Symposium on Modeling and Optimization in Mobile, Ad-Hoc and Wireless Networks - WiOpt 2005*, pages 41–46. IEEE Computer Society, 2005.

[Mer89] Ralph C. MERKLE : One way hash functions and des. *In Advances in Cryptology - CRYPTO '89*, volume 435 de *Lecture Notes in Computer Science*, pages 428–446. Springer, 1989.

[Mil67] Stanley MILGRAM : The Small World Problem. *Psychology Today*, 2:60–67, 1967.

[MK92] M. MATSUMOTO et Y. KURITA : Twisted GFSR generators. *ACM Trans. Model. Comput. Simul.*, 2(3):179–194, 1992.

[MK02] Muriel MÉDARD et Ralf KOETTER : Beyond Routing : An Algebraic Approach to Network Coding. *In IEEE INFOCOM 2002*. IEEE, 2002.

[MKK00] J.L. MASSEY, G.H. KHACHATRIAN et M. K. KUREGIAN : SAFER ++. *In First Open Nessie Workshop*, Leuven, Belgique. I.S.T., téléchargeable à l'adresse `http://www.cryptonessie.org/`, 2000.

[MMO85] S.M. MATYAS, C.H. MEYER et J. OSEAS : Generating strong one-way functions with cryptographic algorithm. IBM Technical Disclosure Bulletin 27, 1985. 5658–5659.

[MN98] M. MATSUMOTO et T. NISHIMURA : Mersenne twister : A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Trans. Model. Comput. Simul.*, 8(1):3–30, 1998.

[MNPN+09] Krystian MATUSIEWICZ, María NAYA-PLASENCIA, Ivica NIKOLIC, Yu SASAKI et Martin SCHLÄFFER : Rebound attack on the full lane compression function. *In Advances in Cryptology - ASIACRYPT 2009*, volume 5912 de *Lecture Notes in Computer Science*, pages 106–125. Springer, 2009.

[MNPP11] Marine MINIER, María NAYA-PLASENCIA et Thomas PEYRIN : Analysis of Reduced-SHAvite-3-256 v2. *In Fast Software Encryption - FSE 2011*, volume 6733 de *Lecture Notes in Computer Science*, pages 68–87. Springer, 2011.

[MP] Marine MINIER et Raphael C.-W. PHAN : Open Key Distinguishers for the AES. soumis.

[MP09] Marine MINIER et Benjamin POUSSE : Improving integral cryptanalysis against rijndael with large blocks. *CoRR*, abs/0910.2153, 2009.

[MPP09] Marine MINIER, Raphael C.-W. PHAN et Benjamin POUSSE : Distinguishers for ciphers and known key attack against rijndael with large blocks. *In Progress in Cryptology - AFRICACRYPT 2009*, volume 5580 de *Lecture Notes in Computer Science*, pages 60–76. Springer, 2009.

[MPP10] Marine Minier, Raphael C.-W. Phan et Benjamin Pousse : Integral Distinguishers of Some SHA-3 Candidates. *In Cryptology and Network Security - CANS 2010*, volume 6467 de *Lecture Notes in Computer Science*, pages 106–123. Springer, 2010.

[MPRS09] Florian Mendel, Thomas Peyrin, Christian Rechberger et Martin Schläffer : Improved cryptanalysis of the reduced grøstl compression function, ECHO permutation and AES block cipher. *In Selected Areas in Cryptography - SAC 2009*, volume 5867 de *Lecture Notes in Computer Science*, pages 16–35. Springer, 2009.

[MRH04] Ueli M. Maurer, Renato Renner et Clemens Holenstein : Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. *In Theory of Cryptography, First Theory of Cryptography Conference, TCC 2004*, volume 2951 de *Lecture Notes in Computer Science*, pages 21–39. Springer, 2004.

[MRST09] Florian Mendel, Christian Rechberger, Martin Schläffer et Søren S. Thomsen : The Rebound Attack : Cryptanalysis of Reduced Whirlpool and Grøstl. *In Fast Software Encryption - FSE 2009*, volume 5665 de *Lecture Notes in Computer Science*, pages 260–276. Springer, 2009.

[MRT04] Grzegorz Mrugalski, Janusz Rajski et Jerzy Tyszer : Ring generators - new devices for embedded test applications. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 23(9):1306–1320, 2004.

[MS01] Curt Schurgers Mani et Mani B. Srivastava : Energy efficient routing in wireless sensor networks. *In Military Communications Conference Proceedings on Communications for Network-Centric Operations : Creating the Information Force*, volume 1, pages 357–361, 2001.

[Nat77] National Bureau of Standards (États Unis) : Data Encryption Standard. *Federal Information Processing Standard*, 1977. Publication 46.

[Nat07] National Institute of Standards and Technology : Announcing Request for Candidate Algorithm Nominations for a New Cryptographic Hash Algorithm (SHA-3) Family. *Federal Register*, 27(212):62212–62220, November 2007. `http://csrc.nist.gov/groups/ST/hash/documents/FR_Notice_Nov07.pdf`.

[NES01] NESSIE : Nessie phase 1 : selection of primitives. `https://www.cryptonessie.org/`, 2001.

[Neu96] J.-P. Neuville : *Le contrat de confiance : étude des mécanismes de coopération dans le partenariat industriel autour de deux grands constructeurs automobiles européens.* Thèse de doctorat, IEP de Paris (sous la direction de E. Friedberg), 1996.

[Nie95] H. Niederreiter : The multiple-recursive matrix method for pseudorandom number generation. *Finite Fields Appl.*, 1(1):3–30, 1995.

[NSSP04] James Newsome, Elaine Shi, Dawn Xiaodong Song et Adrian Perrig : The sybil attack in sensor networks : analysis & defenses. *In Proceedings of the Third International Symposium on Information Processing in Sensor Networks - IPSN 2004*, pages 259–268. ACM, 2004.

[NV09] Dhiraj Nitnaware et Ajay Verma : Energy evaluation of proactive and reactive protocol for manet under on/off source traffic. *In Proceedings of the International*

Conference on Advances in Computing, Communication and Control - ICAC3 '09, pages 451–455. ACM, 2009.

[oC93] U.S. Department of Commerce : FIPS 180 : Secure Hash Standard. Federal Information Processing Standards Publication, N.I.S.T., 1993.

[oC95] U.S. Department of Commerce : FIPS 180-1 : Secure Hash Standard (SHS). Federal Information Processing Standards Publication, N.I.S.T., 1995.

[oST01] National Institute of Standards et Technology : AES competition, Call for Block Cipher Primitives 1997-2001. `http://csrc.nist.gov/archive/aes/`.

[OW85] L. Ozarow et A. Wyner : Wire-tap channel II. *In Advances in Cryptology - EUROCRYPT 1984*, pages 33–50. Springer, 1985.

[Pai99] Pascal Paillier : Public-key cryptosystems based on composite degree residuosity classes. *In Advances in Cryptology - EUROCRYPT 1999*, volume 1592 de *Lecture Notes in Computer Science*, pages 223–238. Springer, 1999.

[PCTS02] Adrian Perrig, Ran Canetti, J.D. Tygar et Dawn Song : The tesla broadcast authentication protocol. *RSA CryptoBytes*, 5(Summer), 2002.

[PL05a] François Panneton et Pierre L'Ecuyer : On the xorshift random number generators. *ACM Trans. Model. Comput. Simul.*, 15(4):346–361, 2005.

[PL05b] Pascal Pons et Matthieu Latapy : Computing communities in large networks using random walks. *In Computer and Information Sciences - ISCIS 2005*, volume 3733 de *Lecture Notes in Computer Science*, pages 284–293. Springer, 2005.

[PM08] Benjamin Pousse et Marine Minier : On the security of fcsr-based pseudorandom generators. SASC 2008 - Stream Ciphers Revisited, February 2008. Special Workshop hosted by the ECRYPT Network of Excellence.

[Pou10] Benjamin Pousse : *Design et cryptanalyse de chiffrements à flot*. Thèse de doctorat, Université de Limoges, 2010. disponible en ligne `http://epublications.unilim.fr/theses/2010/pousse-benjamin/pousse-benjamin.pdf`.

[PPG05] Bryan Parno, Adrian Perrig et Virgil D. Gligor : Distributed detection of node replication attacks in sensor networks. *In 2005 IEEE Symposium on Security and Privacy - S&P 2005*, pages 49–63. IEEE Computer Society, 2005.

[Pro01] Third Generation PartnerShip Project : 3GPP TS 35.202 - Specification of the 3GPP Confidentiality and Integrity algorithms - Document 2 : KASUMI specification. `http://www.3gpp.org/ftp/Specs/html-info/35202.htm`, 2001.

[QHC06] Daniele Quercia, Stephen Hailes et Licia Capra : Tata : Towards anonymous trusted authentication. *In Trust Management, 4th International Conference - iTrust 2006*, volume 3986 de *Lecture Notes in Computer Science*, pages 313–323. Springer, 2006.

[RCC+04] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto et D. Parisi : Defining and identifying communities in networks. *Proceedings of the National Academy of Science of the United States of America, PNAS*, 101(9):2658–2663, 2004.

[Riv90] Ronald L. Rivest : The MD4 Message Digest Algorithm. *In Advances in Cryptology - CRYPTO '90*, volume 537 de *Lecture Notes in Computer Science*, pages 303–311. Springer, 1990.

[Riv92a] R. Rivest : The RC4 encryption algorithm. RSA Data Security, 1992.

[Riv92b] Ronald L. Rivest : The MD5 Message Digest Algorithm. Request for Comments (RFC 1320), 1992. Activities Board, Internet Privacy Task Force.

[Rog89] Y. Roggeman : Varying feedback shift registers. *In Advances in Cryptology - EUROCRYPT 1989*, volume 434 de *Lecture Notes in Computer Science*, pages 670–679. Springer, 1989.

[SA99] Frank Stajano et Ross J. Anderson : The resurrecting duckling : Security issues for ad-hoc wireless networks. *In Security Protocols Workshop, 7th International Workshop - SPW 1999*, volume 1796 de *Lecture Notes in Computer Science*, pages 172–194. Springer, 1999.

[Sas10] Yu Sasaki : Known-key attacks on rijndael with large blocks and strengthening *shiftrow* parameter. *In Advances in Information and Computer Security - IWSEC 2010*, volume 6434 de *Lecture Notes in Computer Science*, pages 301–315. Springer, 2010.

[SB02] S. D. Servetto et G. Barrenechea : Constrained random walks on random graphs : routing algorithms for large scale wireless sensor networks. *In Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications - WSNA 2002*, pages 12–21. ACM, 2002.

[SC88] B. J. M. Smeets et W. G. Chambers : Windmill generators : A generalization and an observation of how many there are. *In Advances in Cryptology - EUROCRYPT 1988*, volume 434 de *Lecture Notes in Computer Science*, pages 325–330, 1988.

[Sco88] Mike Scott : Miracl - a multiprecision library. *The C Users Journal*, 6:76–80, 1988. `http://www.shamus.ie/`.

[Sha49] C. Shannon : Communication theory of secrecy systems. *Bell System Technical Journal, Vol 28, pp. 656-715*, October 1949.

[SHJ09] Paul Stankovski, Martin Hell et Thomas Johansson : An Efficient State Recovery Attack on X-FCSR-256. *In Fast Software Encryption - FSE 2009*, volume 5665 de *Lecture Notes in Computer Science*, pages 23–37. Springer, 2009.

[SHJar] Paul Stankovski, Martin Hell et Thomas Johansson : An Efficient State Recovery Attack on the X-FCSR Family of Stream Ciphers. *J. Cryptology*, to appear.

[SPR+09] Reza Shokri, Marcin Poturalski, Gael Ravot, Panos Papadimitratos et Jean-Pierre Hubaux : A practical secure neighbor verification protocol for wireless sensor networks. *In Proceedings of the second ACM conference on Wireless network security - WISEC 2009*, pages 193–200. ACM, 2009.

[SS10] Damien Stehlé et Ron Steinfeld : Faster fully homomorphic encryption. *In Advances in Cryptology - ASIACRYPT 2010*, volume 6477 de *Lecture Notes in Computer Science*, pages 377–394. Springer, 2010.

[SV10] Nigel P. Smart et Frederik Vercauteren : Fully homomorphic encryption with relatively small key and ciphertext sizes. *In Public Key Cryptography - PKC 2010*, volume 6056 de *Lecture Notes in Computer Science*, pages 420–443. Springer, 2010.

[SY11] Yu Sasaki et Kan Yasuda : Known-Key Distinguishers on 11-Round Feistel and Collision Attacks on Its Hashing Modes. *In Fast Software Encryption - FSE 2011*,

volume 6733 de *Lecture Notes in Computer Science*, pages 397–415. Springer, 2011.

[TCG91] Anne Tardy-Corfdir et Henri Gilbert : A Known Plaintext Attack of FEAL-4 and FEAL-6. *In Advances in Cryptology - CRYPTO '91*, volume 576 de *Lecture Notes in Computer Science*, pages 172–181. Springer, 1991.

[Toh02] C.K. Toh : *Ad hoc mobile wireless networks : protocols and systems*. Prentice Hall PTR, 2002.

[Vau03] Serge Vaudenay : Decorrelation : A theory for block cipher security. *J. Cryptology*, 16(4):249–286, 2003.

[Wag99] David Wagner : The boomerang attack. *In Fast Software Encryption - FSE '99*, volume 1636 de *Lecture Notes in Computer Science*, pages 156–170. Springer, 1999.

[Wag02] David Wagner : A generalized birthday problem. *In Advances in Cryptology - CRYPTO 2002*, volume 2442 de *Lecture Notes in Computer Science*, pages 288–303. Springer, 2002.

[Wag04] David Wagner : Resilient aggregation in sensor networks. *In Proceedings of the 2nd ACM Workshop on Security of ad hoc and Sensor Networks - SASN 2004*, pages 78–87. ACM, 2004.

[WSS03] A.D. Wood, J.A. Stankovic et S.H. Son : Jam : a jammed-area mapping service for sensor networks. *24th IEEE Real-Time Systems Symposium - RTSS 2003*, pages 286–297, 2003.

[WY05] Xiaoyun Wang et Hongbo Yu : How to Break MD5 and Other Hash Functions. *In Advances in Cryptology - EUROCRYPT 2005*, volume 3494 de *Lecture Notes in Computer Science*, pages 19–35. Springer, 2005.

[WYY05a] Xiaoyun Wang, Yiqun Lisa Yin et Hongbo Yu : Finding Collisions in the Full SHA-1. *In Advances in Cryptology - CRYPTO 2005*, volume 3621 de *Lecture Notes in Computer Science*, pages 17–36. Springer, 2005.

[WYY05b] Xiaoyun Wang, Hongbo Yu et Yiqun Lisa Yin : Efficient Collision Search Attacks on SHA-0. *In Advances in Cryptology - CRYPTO 2005*, volume 3621 de *Lecture Notes in Computer Science*, pages 1–16. Springer, 2005.

[XV07] Dawei Xia et Natalija Vlajic : Near-optimal node clustering in wireless sensor networks for environment monitoring. *In Proceedings of the 21st International Conference on Advanced Networking and Applications - AINA 2007*, pages 632–641. IEEE Computer Society, 2007.

[YHC07] Eiko Yoneki, Pan Hui et Jon Crowcroft : Visualizing community detection in opportunistic networks. *Proceedings of the second workshop on Challenged networks - CHANTS 2007*, pages 93–96, 2007.

[YLCZ05] R. W. Yeung, S.-Y. R. Li, N. Cai et Z. Zhang : *Network Coding Theory*. now Publishers, 2005.

[YWRG08] Zhen Yu, Yawen Wei, Bhuvaneswari Ramkumar et Yong Guan : An Efficient Signature-Based Scheme for Securing Network Coding Against Pollution Attacks. *In 27th IEEE International Conference on Computer Communications - INFOCOM 2008*, pages 1409–1417. IEEE, 2008.

[YWRG09] Zhen Yu, Yawen Wei, B. Ramkumar et Yong Guan : An Efficient Scheme for Securing XOR Network Coding against Pollution Attacks. *In IEEE INFOCOM 2009*, pages 406–414, Rio de Janeiro, Brasil, April 2009.

[ZLM09] Wassim Znaidi, Cédric Lauradoux et Marine Minier : Aggregated Authentification (AMAC) using Universal Hash Functions. *In International ICST Conference on Security and Privacy in Communication Networks - SecureComm 2009*, volume 17 de *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 248–264. Springer, 2009.

[ZM10] Wassim Znaidi et Marine Minier : Key management and access control scheme for wsns. *the Telecommunication Systems Journal*, accepté en Septembre 2010. à paraître.

[ZMB08a] Wassim Znaidi, Marine Minier et Jean-Philippe Babau : An Ontology for Attacks in Wireless Sensor Networks. Rapport de recherche RR-6704, INRIA, 2008.

[ZMB08b] Wassim Znaidi, Marine Minier et Jean-Philippe Babau : Detecting wormhole attacks in wireless networks using local neighborhood information. *In Proceedings of the IEEE 19th International Symposium on Personal, Indoor and Mobile Radio Communications - PIMRC 2008*, pages 1–5. IEEE, 2008.

[ZMU09] Wassim Znaidi, Marine Minier et Stéphane Ubéda : Hierarchical node replication attacks detection in wireless sensors networks. *In Proceedings of the IEEE 20th International Symposium on Personal, Indoor and Mobile Radio Communications - PIMRC 2009*, pages 82–86. IEEE, 2009.

[Zna10] Wassim Znaidi : *Quelques propositions de solutions pour la sécurité des réseaux de capteurs sans fil.* Thèse de doctorat, INSA de Lyon, 2010.

[ZZLM11] Yuanyuan Zhang, Wassim Znaidi, Cédric Lauradoux et Marine Minier : Flooding attacks against network coding and countermeasures. *In International Conference on Network and System Security - NSS 2011*. IEEE, 2011.

Annexe A

Articles présentés dans la première partie

Some results on FCSR automata with applications to the security of FCSR-based pseudorandom generators

François Arnault, Thierry P. Berger and Marine Minier

***Abstract*—This article describes new theoretical results concerning the general behavior of a FCSR (Feedback with Carry Shift Register)automaton that allow to better understand how the initial parameters must be chosen to use this automaton as a basic block of a filtered stream cipher. The results demonstrated here especially concern the structure of the transition graph of an FCSR automaton and the number of iterations of the FCSR transition function required to reach the main part of the graph. A potential linear weakness and a easy way to discard the corresponding attack are also given.**

Keywords: Stream ciphers, FCSR, 2-adic expansion, transition function graph.

I. Background on FCSR automata

The Feedback with Carry Shift Registers were introduced first by Klapper and Goresky in [11]. In [1], T. Berger and F. Arnault proposed to use them as the core of a filtered stream cipher. We first recall how a FCSR automaton works. For more details, the reader could refer to [8], [1], [4].

A. Representation of eventually periodic binary sequences with 2-adic numbers

First, we will recall briefly some basic properties of 2-adic numbers. For a more theoretical approach the reader can refer to [4], [7], [9], [11], [12].

A 2-adic integer is formally a power series $s = \sum_{n=0}^{\infty} s_n 2^n$, $s_n \in \{0, 1\}$. Such a series does not always converge in the classical sense. However, it can be considered as a formal object. Actually, this series always converges if we consider the 2-adic topology. The set of 2-adic integers is denoted by $\mathbb{Z}_2$. Addition and multiplication in $\mathbb{Z}_2$ can be performed by reporting the carries to the higher order terms, i.e. $2^n + 2^n = 2^{n+1}$ for all $n \in \mathbb{N}$. If there exists an integer N such that $s_n = 0$ for all $n \geq N$, then s is a positive integer. Moreover, every odd integer q has an inverse in $\mathbb{Z}_2$ which can be computed by the formula $q^{-1} = \sum_{n=0}^{\infty} q'^n$, where $q = 1 - q'$.

The following theorem gives a complete characterization of eventually periodic 2-adic binary sequences in terms of 2-adic integers (see [9] for the proof).

This work was partially supported by the french National Agency of Research: ANR-06-SETI-013

François Arnault and Thierry P. Berger are with XLIM-DMI, UMR CNRS 6171, Université de Limoges, 123 avenue A. Thomas, 87060, Limoges cedex, France, arnault@unilim.fr thierry.berger@unilim.fr

Marine Minier is with CITI laboratory, INSA de Lyon, 21 Avenue Jean Capelle, F-69621 Villeurbanne Cedex, FRANCE, marine.minier@insa-lyon.fr

Theorem 1: Let $S = (s_n)_{n\in\mathbb{N}}$ be a binary sequence and let $s = \sum_{n=0}^{\infty} s_n 2^n$ be the associated 2-adic integer. The sequence S is eventually periodic if and only if there exist two numbers p and q in $\mathbb{Z}$, q odd, such that $s = p/q$. Moreover, S is strictly periodic if and only if $pq \leq 0$ and $|p| \leq |q|$.

An important fact for applications is that the period of the rational number p/q is known (cf. [9]):

Theorem 2: Let S be an eventually periodic binary sequence, and let $s = p/q$, with q odd and p and q coprime, be the corresponding 2-adic number in its rational representation. The period of S is the order of 2 modulo q, i.e., the smallest integer T such that $2^T \equiv 1 \pmod{q}$.

The period T is always less or equal to $|q|-1$. If q is prime, then T divides $|q| - 1$. If $T = |q| - 1$, the corresponding sequence S is called a ℓ-sequence. For more details on ℓ-sequences see [11], [8].

B. FCSR Automaton in Galois mode

Feedback with Carry Shift Registers (FCSR) automata was firstly introduced by Klapper and Goresky in [11]. This first version corresponds to the analog of LFSR in Fibonnacci mode. In [9], they introduced a Galois' version of FCSR which seems more suitable for practical implementation. In this paper, we describe only FCSR in Galois mode.

A FCSR automaton is defined using an odd negative connection integer q of binary size n: $2^n < -q < 2^{n+1}$.

Let d be the positive integer $d = (1 - q)/2$ and let $d = \sum_{i=0}^{n-1} d_i 2^i$ its binary expansion. Note that $d_{n-1} = 1$. We denote by $J = \{i : 0 \leq i \leq n - 2,\ d_i \neq 0\}$ the support of d but without the $n - 1$ position. If ℓ denotes the cardinality of J, we arrange J in the following way: $J = \{i_1, \ldots, i_\ell\}$, with $i_j < i_{j+1},\ \forall j \in [1..\ell]$.

The automaton then consists of two registers:

- A main register M composed of n cells denoted by m_i $(0 \leq i \leq n - 1)$.
- A carries register C composed of ℓ cells denoted by c_{i_j} $(1 \leq j \leq \ell)$.

For simplicity, we consider that the carries register C contains n cells c_i $(0 \leq i \leq n - 1)$, with $c_i = 0$ if $d_i = 0$. However, the true size of a FCSR automaton is $n + \ell$ cells.

The transition function of the registers at time t can be written at the cell level:

$m_i(t + 1) = m_{i+1}(t) \oplus d_i c_i(t) \oplus d_i m_0(t)$

$c_i(t + 1) = d_i\,(m_{i+1}(t)c_i(t) \oplus c_i(t)m_0(t) \oplus m_0(t)m_{i+1}(t))$

where $\oplus$ denotes the bitwise XOR.

C. Properties of the transition function of a FCSR automaton

To each state of the main register M and of the carries register C, we can associate the following integers m and c (also denoted by $m(t)$ and $c(t)$ at time t): $m = \sum_{i=0}^{n-1} m_i 2^i$ and $c = \sum_{i=0}^{n-2} c_i 2^i$. These integers are called the contents of M and C. If the main register and the carries register contain the integer $m(t)$ and $c(t)$ at time t, we say that the automaton is in state $(m(t), c(t))$. Let $p(t)$ denote the integer $m(t)+2c(t)$.

Lemma 1 ([10]*):* We have $2p(t+1) \equiv p(t) \bmod q$.

Proposition 1 ([9]*):* If a FCSR automaton is in the state $(m, c) = (m(0), c(0))$ at time $t = 0$, it computes the 2-adic expansion of the rational 2-adic number p/q (where $p = m + 2c$), which is produced by the cell m_0: $p/q = \sum_{t\geq 0} m_0(t)2^t$.

Distinct initial states can produce the same sequence.

Definition 1: Two states (m, c) and (m', c') are said equivalent if they satisfy $m + 2c = m' + 2c'$, i.e. $p = p'$.

As a direct consequence of Proposition 1, we have:

Proposition 2: Two states (m, c) and (m', c') are equivalent if and only if they compute the same 2-adic fraction p/q, i.e. the sequences observed in the cell m_0 are equal and correspond to the 2-adic expansion of p/q, with $p = m + 2c$.

D. Structure of the graph of a FCSR automaton

To each binary automaton with k cells, we can associate its transition graph which is defined as follows: The nodes are the 2^k possible states. There exists an edge from a state A to a state B iff the state B is the image of A by the transition function of the automaton.

There is a bijection between the cyclotomic cosets $C_p = \{p2^i \bmod q\}$ and the connected components of the graph of the FCSR automaton with connection integer q. A state (m, c) belongs to the component associated with C_p, where $p = m + 2c$. This is a consequence of the fact that two states are equivalent if and only if they eventually converge to a same state after the same finite number of iterations. This result will be proved in Section II-A Proposition 5.

In particular, the graph of a FCSR automaton has always two single node connected components associated with $p = 0$ and $p = -q$. They correspond to the invariant states $(0, 0)$ and $(2^n - 1, d - 2^{n-1})$ (in the latter, all the $n + \ell$ cells contain the bit value 1).

If the order of 2 modulo q is exactly $T = |q| - 1$, i.e. the automaton generates ℓ-sequences, the graph contains only one more component with $2^{n+l} - 2$ points, considered as a main cycle of length $|q| - 1$ to which many tails converge (see [1] for more details).

Definition 2: We say that a FCSR automaton with connection integer q is optimal if the order of 2 modulo q is exactly $T = |q| - 1$.

In Subsection II-B, we will bound the lengths of the tails in the graph of any FCSR automaton.

E. Sequences produced by the main register of a FCSR

Now we will look at the sequences of bits produced by the cells of the main register. These sequences are denoted $M_i = (m_i(t))_{t\in\mathbb{N}}$, for $0 \leq i \leq n-1$. The following theorem was proved in [4].

Theorem 3 ([4]*):* Consider a FCSR automaton with negative connection integer $q = 1-2d$. Let n be the bitlength of d. Then, for all i such that $0 \leq i \leq n-1$, there exists an integer p_i such that the sequence M_i observed in the cell number i of the main register is the 2-adic expansion of p_i/q. Moreover, the integers p_i satisfy the following recurrence relation:

$$p_i = q\big(m_i(0) + 2c_i(0)\big) + 2p_{i+1} + 2d_i p_0$$

with the convention $c_i(t) = 0$ when $d_i = 0$.

From now, we will use the following notations:

- Content of a cell (cell level): $m_i = m_i(0)$, $c_i = c_i(0)$.
- Content of a whole register (integer level): $m = m(0)$, $c = c(0)$.
- Observed sequences, in a specific cell of the main register, from time t_0: $M_i(t_0) = (m_i(t))_{t\geq t_0}$ for $0 \leq i \leq n-1$.

In particular $M_i(0) = M_i$ for $0 \leq i \leq n-1$.

- The 2-adic fractions corresponding to these sequences: $M_i(t_0) = p_i(t_0)/q$, i.e. $p_i(t_0)$ is the integer satisfying the relation: $p_i(t_0) = q \times \sum_{t\geq t_0} m_i(t)2^i$.

Note that $p_i = p_i(0)$ for $0 \leq i \leq n-1$ and $p = p_0 = p_0(0)$.

F. Some properties of 2-adic rational numbers

We present in this section some elementary properties of the 2-adic sequences, required to demonstrate results of the next section. These results are relatively simple and essentially known (see [7], [11], [8] for further details).

First, we introduce the following notations:

- $A_q = \{p/q | 0 \leq p \leq -q\}$ the set of the 2-adic periodic sequences with a denominator equal to q: $2^n < -q < 2^{n+1}$.
- $A_q^* = \{p/q | 0 < p < -q\}$ and $p/q = \sum_{i=0}^{\infty} a_i 2^i$
- $\mathbb{N}_k = \{0, 1, \cdots, 2^k - 1\} = \{\sum_{i=0}^{k-1} a_i 2^i, a_i = 0 \text{ or } 1\}$

Proposition 3: Remember that $2^n < -q \leq 2^{n+1}$. We have the following properties:

- The map f_n from A_q^* to $\mathbb{N}_n$ defined by $f_n(p/q) = p/q \bmod 2^n$ is surjective.
- The map f_{n+1} from A_q to $\mathbb{N}_{n+1}$ defined by $f_n(p/q) = p/q \bmod 2^{n+1}$ is injective.

Corollary 1: For any odd q satisfying $2^n < -q \leq 2^{n+1}$ and any p, $0 < p < |q|$, there is no sequence of $n+1$ consecutive zeros and there is no sequence of $n+1$ consecutive ones in the 2-adic expansion of p/q.

Corollary 2: If the order of 2 modulo q is maximal (i.e. equals to $-q-1$), then the 2^n sequences with n consecutive bits appear at least one time and at most two times in a period of the binary expansion of any $p/q \in A_q^*$.

Corollary 3: If the order of 2 modulo q is maximal (i.e. equals to $-q-1$) and if $p/q \in A_q^*$, then the subsequence with n consecutive bits all equal to 0 (resp. all equal to 1) appears one and only one time in the period of the binary expansion of p/q.

II. New results on FCSR automata

We present in this section some important results concerning the number of transitions necessary to reach a cycle (the cycle when we consider an optimal FCSR) and the entropy of a FCSR automaton. Indeed, while the total number of states of an FCSR automaton is $2^{n+\ell}$, a cycle is composed of T states (where T is the order of 2 modulo q). The other states are

distributed on tails or trees which converge quickly to this cycle. Thus, we want to determine an upper bound on the lengths of tails that are attached to a cycle (or the cycle).

In the case of an optimal FCSR (that could be used in a stream cipher construction), the cycle is composed of $|q|-1$ states (with always $2^n \leq |q|-1 < 2^{n+1}$). To guarantee some properties of FCSR based stream ciphers, we don't want the optimal FCSR to output pseudorandom data before it has reached the cycle.

A. Explicit determination of sequences M_i

In this paragraph, we will determine the exact values of each p_i (or $p_i(t)$) defined in Theorem 3. We always suppose that the initial state of the automaton is not a fixed point of the transition function, i.e. $0 < p < |q|$.

To simplify the presentation, we suppose from now that $t_0 = 0$ without loss of generality. We also need to introduce the following notations, for $0 \leq i \leq n-1$:
$d^{(i)} = \sum_{j=0}^{i-1} d_j 2^j, \quad \delta^{(i)} = \sum_{j=i}^{n-1} d_j 2^{j-i}$,
so that $d = d^{(i)} + 2^i \delta^{(i)}$,
$u^{(i)} = \sum_{j=0}^{i-1} (m_j + 2c_j) 2^j, \quad \nu^{(i)} = \sum_{j=i}^{n-1} (m_j + 2c_j) 2^{j-i}$,
so that $p = u^{(i)} + 2^i \nu^{(i)}$.

Proposition 4: With the above notations, we have the following relation:

$$p_i = q\nu^{(i)} + 2p\delta^{(i)}. \tag{1}$$

Proof: We perform the proof by induction on i. For $i = 0$ we have $p_0 = p$, $\delta^{(0)} = d$, and $\nu^{(0)} = p$. Hence, $q\nu^{(0)} + 2p\delta^{(0)} = p(q + 2d) = p = p_0$.

Suppose now that the relation (1) is true for i. Let us prove that the relation stays true at step $i+1$. From Theorem 3, we have $p_i = q(m_i(0) + 2c_i(0)) + 2p_{i+1} + 2d_i p_0$. We also can write $\delta^{(i)} = d_i + 2\delta^{(i+1)}$ and $\nu^{(i)} = m_i + 2c_i + 2\nu^{(i+1)}$. Using the induction hypothesis, we obtain:

$$\begin{aligned} q\nu^{(i)} + 2p\delta^{(i)} = p_i &= q(m_i(0) + 2c_i(0)) + 2p_{i+1} + 2d_i p \\ &= q(\nu^{(i)} - 2\nu^{(i+1)}) + 2p_{i+1} + 2(\delta^{(i)} - 2\delta^{(i+1)})p. \end{aligned}$$

Canceling $q\nu^{(i)}$ and $2p\delta^{(i)}$ on both sides we obtain $2p_{i+1} = 2q\nu^{(i+1)} + 4p\delta^{(i+1)}$. This is the relation (1) for $i+1$: $p_{i+1} = q\nu^{(i+1)} + 2p\delta^{(i+1)}$ and this concludes our proof. ■

Corollary 4: Assume that (m, c) and (m', c') are two equivalent states, and that p_i/q and p'_i/q are the respective fractions whose expansion is given by the cells m_i and m'_i. Then we have $p_i \equiv p'_i \pmod q$. Moreover, if (m', c') is in a cycle, then $0 \leq p'_i < |q|$.

Proof: By Proposition 2, the result is true for the special case $i = 0$. By Proposition 4, we have $p_i \equiv 2p\delta^{(i)}$ and $p'_i \equiv 2p'\delta^{(i)}$ modulo q for any i. We obtain $p'_i \equiv 2p'\delta^{(i)} = 2p\delta^{(i)} \equiv p_i$ modulo q, and this is the first claim. If (m', c') is in a cycle then the sequence p_i/q is periodic so we have $0 \leq p'_i < q$ from Theorem 1. This proves the second claim. ■

The following proposition revisits the notion of equivalent states in view of the transition function. The proof given here concerns the general (optimal or not) case.

Proposition 5: Two states are equivalent if and only if they eventually converge to a same state after the same number of iterations.

Proof: Assume that the states (m, c) and (m', c') are equivalent. There exists a positive integer k such that the states obtained by applying k times the transition function to the initial states produce the same state on a cycle. At this point, for each i, from Definition 3 the sequences $(m_i(t))_{t \geq k}$ and $(m'_i(t))_{t \geq k}$ are both periodic. By Theorem 1, we obtain $0 \leq p_i(k) < -q$ and $0 \leq p'_i(k) < -q$. By Corollary 4, we have $p_i(k) \equiv p'_i(k)$ modulo q, so $p_i(k) = p'_i(k)$ for each i. Using that $m(k) + 2c(k) = p(k) = p'(k) = m'(k) + 2c'(k)$, we see that $c(k) = c'(k)$ also. Hence, the two states reached at step k are equal.

The converse is easy, using Lemma 1. ■

B. Maximum length of the tails of the FCSR graph

The properties described above are useful to compute the number of transitions required to reach a cycle (the cycle, in the case of an optimal FCSR). Consider a FCSR automaton with connection integer q. We say that the automaton is synchronized if it has reached a state on a cycle. We consider here each binary sequence M_i of the main register.

Definition 3: We say that the cell m_i is synchronized at time t if the sequence of the $(m_i(t+j))_{j \geq 0}$ values is periodic, i.e. $m_i(t+j) = m_i(t+j+T),\ \forall j \geq 0$, for some period $T > 0$.

Lemma 2: The state (m, c) belongs to a cycle if and only if all the cells of the main register are synchronized.

Proof: Assume that all cells of the main register are synchronized. By Proposition 5, The state (m, c) is equivalent to a state (m', c') which belongs to a cycle. Starting from this state (m', c'), the content of each cell of the main register is a sequence $M'_i = (m'_i(t))_{t \geq 0}$. The values $M_i(t)$ and $M'_i(t)$ are equal as soon as t is large enough. But the sequences M_i and M'_i are both periodic so $M_i(t)$ and $M'_i(t)$ are equal for all $t \geq 0$. As a consequence, we obtain $m = m'$. As $m + 2c = m' + 2c'$, we obtain also $c = c'$. This shows that the state (m, c) belongs to a cycle. The converse is clear. ■

Proposition 6: The cell m_i is synchronized at time t if and only if we have $0 < p_i(t) < |q|$.

Proof: The cell m_i is synchronized at time t if and only if the sequence $M_i(t)$ is periodic, which is equivalent to $0 < p_i(t) < |q|$ (cf. Theorem 1). ■

Lemma 3: Adding or subtracting a positive integer $b = \sum_{i=0}^{m-1} b_i 2^i$ of bit length m to the 2-adic fraction p/q when $0 < p < |q|$ affects at most the $m + n + 1$ first bits.

Proof: Consider the addition case. Denote $(r(t))_{t \geq 0}$ the 2-adic expansion of the fraction p/q. By Corollary 1, there is at least one integer k such that $m \leq k \leq m + n$ with $r(k) = 0$. Adding 2^k to p/q would change the bit $r(k)$ and leave the other terms of the sequence unchanged. But $b < 2^m \leq 2^k$. So that, adding b changes only some bits of lower weight $r(j)$ with $j \leq k$. The subtraction case is similar. ■

Hence, a bound on the number of iterations required for each cell to synchronize, will provide a general bound for the synchronization of the whole automaton.

Lemma 4: $\forall i$ such that $0 \leq i \leq n-1$, we have: $6q < p_i < -8q$.

Proof: For $i = 0$, we have $0 \leq p_0 = p < -q$ so the claim is true in that case. Assume now that $i > 0$. From the

definitions given above, we have:
$0 \leq \delta^{(i)} < 2^{n-i}, \qquad 0 \leq d^{(i)} < 2^i,$
$0 \leq \nu^{(i)} < 3 \times 2^{n-i}, \quad 0 \leq u^{(i)} < 3 \times 2^i.$
By Proposition 4 we have $p_i = \nu^{(i)} - 2\nu^{(i)}d^{(i)} + 2u^{(i)}\delta^{(i)}$. Using the above inequalities, we get
$-3 \times 2^{n+1} < -2\nu^{(i)}d^{(i)} \leq p_i$
$\leq \nu^{(i)} + 2u^{(i)}\delta^{(i)} < 3 \times (2^{n+1} + 2^{n-1}).$
But, $2^n \leq -q \leq 2^{n+1}$. We finally obtain $6q < p_i < -8q$. ■

The main result of this section is the following theorem:

Theorem 4: Suppose that a FCSR automaton with connection integer q starts from the state (m, c). Then it will be synchronized after at most $n + 4$ iterations (n denotes the length of the main register). More generally, the lengths of the tails of the graph of a FCSR automaton are at most $n+4$.

Proof: There exists a state (m', c') which is equivalent to (m, c) and belongs to a cycle. As the sequence of bits obtained in the cell m_i is periodic, we have $0 < p'_i < |q|$. From Corollary 4, we have $p'_i \equiv p_i$ modulo q for each i such that $0 \leq i \leq n - 1$. Also, from Lemma 4, we have $6q < p_i < -8q$. Hence $p_i = p'_i + b_i q$ for small integers b_i satisfying $-7 \leq b_i \leq 6$. The integers $|b_i|$ have a bit length at most 3, so the theorem follows finally from Lemma 3. ■

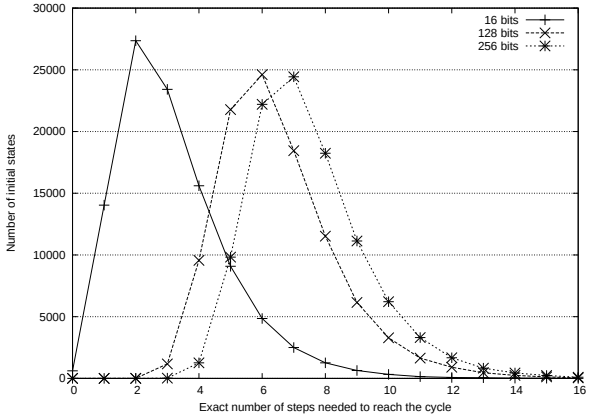


Fig. 1. Number of initial states in function of the exact number of transitions they need to reach the cycle. Three simulations have been done using 100000 random initializations and with three register sizes: 16, 128 and 256 bits.

We have performed an experimentation summed up in Figure 1. We have computed for 100000 random initial values the number of iterations required to reach the cycle for three different q optimal values. We observe that this number of iterations is with high probability well below the bound given in Theorem 4. Thus, the convergence toward the cycle in case of an optimal FCSR is very fast.

This is not surprising. The bound $n + 4$ provided by the Theorem relies on the time to find a zero (or a one) in a particular sequence, and Corollary 1 was used to bound this time to $n + 1$. However, if we consider this sequence as Bernouilli trials, we can expect that the average position of the first 0 is 2. So synchronization is expected to occur after the first 6 steps, for most cells. Moreover, the register is synchronized as soon as all of its cells are. Using further the hypothesis of Bernouilli trials, we can expect that when the size of the register doubles, the synchronization time is increased by 1. This is in accordance with the results of Figure 1, which shows that the size of q does not mainly influence the time required for the synchronization process.

III. A POTENTIAL ATTACK ON F-FCSR PSEUDORANDOM GENERATORS

A. Review on F-FCSR pseudorandom generators

A simple way to construct a pseudorandom generator using FCSRs is to filter the cells of its main register with some boolean functions. If the parameters of the FCSR automaton are correctly chosen, its natural nonlinear behavior makes algebraic attacks infeasible [1], [4].

So, it is not necessary to use a Boolean function with a high nonlinearity. A simple linear function could be used to filter the content of the FCSR main register. This choice appears to be well suited for two main reasons: these functions have an optimal resilience order and offer better resistance to correlation attacks. Moreover, they are efficient and easy to implement for hardware and software applications.

The family of F-FCSR generators (see [1], [2], [4], [5] for example) uses this model: at each iteration t, the bit (or byte in some cases) of output keystream $z(t)$ is obtained by filtering the content of the main register of an optimal FCSR using a linear function.

B. Linear weakness of a FCSR automaton

The potential weakness described here addresses degenerate behavior of the FCSR automaton which can occur when the transition function of the FCSR is linear. This happens when all the cells of the carries register contain a 0 bit and, at the same time, the feedback bit m_0 is also 0. In this case, the transition function becomes a simple circular permutation of the contents of the cells of the main register:

$$m_i(t+1) = m_{(i+1 \bmod n)}(t), \qquad \forall i, 0 \leq i \leq n-1.$$

Suppose that this situation occurs during r consecutive transitions of the FCSR from time t_0. Since a F-FCSR generator filters the FCSR main register using a linear function, the output corresponding to these r iterations linearly depends on the values $m_i(t_0)$ contained in the cells of the main register. Clearly, this fact could be used to design a fast attack. For example, the F-FCSR-16 generator outputs 16 bits at each iteration and the size of the main register contains 256 cells. If $r = 16$, then the linear behavior would allow a system of 256 linear equations in 256 unknowns to be written down. It would be easy to solve this system to recover the complete state of the main register at time t_0, assuming that the 256 linear equations are linearly independent (if they are not, the partial information retrieved is likely to be sufficient to break the generator by an exhaustive search on the space of the remaining possible states).

Let us estimate the *a priori* probability of such a linear behavior to occur. For that purpose, the following lemma is helpful:

Lemma 5: The two conditions:

1) Carries register is 0 and the first r bits of $p_0(t)/q$ are 0.
2) Carries register is 0 and $m_i(t) = 0$ for all i, $0 \leq i < r$.

are equivalent.

Proof: If Condition 2 holds, then the transition function is linear during r steps. Hence $m_i(t+1) = m_{i+1}(t)$ and then $m_0(t+i) = m_i(t)$ so that the r first bits of $p_0(t)/q$ are 0. The converse is also true since the first r bits of $p_0(t)/q$ are the $m_0(t+i)$ for $0 \leq i < r$. ∎

The number of states corresponding to the event $c(t) = 0$ and $m_i(t) = 0$ for all i, $0 \leq i < r$ is 2^{n-r}. Since the total number of states of the automaton is $2^{n+\ell}$, the expected probability of the event equals to $2^{-(\ell+r)}$. As an example, for the stream cipher F-FCSR-16, the chosen parameters were $n = 256$, $\ell = 130$ and $r = 16$, and we obtain an *a priori* probability for that event of 2^{-146}.

C. How to definitively avoid this weakness

Using the following proposition, we have a simple argument to show that this weakness is very easily prevented.

Proposition 7: Let s be the least integer such that $d_s = 1$. Assume that $m_i = c_i = 0$ for all i such that $0 \leq i \leq s$. Then the current state of the automaton is not on the cycle.

Proof: We have $d^{(s+1)} = 2^s$, $u^{(s+1)} = 0$ and $v^{(s+1)} = p/2^{s+1}$. Also, $q = 1 - 2d = 1 - 2(d^{(s+1)} + 2^{s+1}\delta^{(s+1)}) = 1 - 2d^{(s+1)} - 2^{s+2}\delta^{(s+1)}$. Hence $q + 2^{s+2}\delta^{(s+1)} = 1 - 2d^{(s+1)} = 1 - 2^{s+1} < 0$. From Proposition 4, we get $p_{s+1} = qv^{(s+1)} + 2p\delta^{(s+1)} = qp/2^{s+1} + 2p\delta^{(s+1)}p_{s+1} = p/2^{s+1}\big(q + 2^{s+2}\delta^{(s+1)}\big) < 0$. From Proposition 6, the current state of the automaton is not on the cycle. ∎

Consequently, suppose that a FCSR automaton is clocked more than $n+4$ iterations before output is used. If the number r of required equations is greater than s, the situation described in Section III-B cannot occur. This is the case for F-FCSR-16 and F-FCSR-H, the candidates to the second eSTREAM phase for the Profile 2. In these ciphers, the automaton is clocked enough times at each change of IV for the resulting state to be on the cycle. Moreover, $s = 2$ in both cases, and the linear behavior of the automaton cannot occur during more than two consecutive steps.

IV. Conclusion

In this paper, we have given more precise results concerning the general behavior of FCSR automata especially optimal ones. Our main result concerns the number of iterations required to reach a cycle which is bounded by $n+4$ where n represents the bit length of the main register.

In view of the results proved here, we provide some hints in the secure design of pseudorandom generators using an optimal FCSR automaton as a component. First of all, the minimal entropy of an optimal FCSR corresponds to a space of $|q| - 1$ states (with $2^n \leq |q| - 1 < 2^{n+1}$) and is obtained when the automaton has reached a state on the cycle. This is guaranteed after only $n + 4$ iterations. We thus obtain an upper bound on the number of initial transitions needed before output can be used. Moreover, we could initialize a FCSR with $c(0) = 0$ to prevent two equivalent initial states resulting from different keys. In this case, the initial entropy is exactly set to n bits, and it does not decrease when the transition function is applied.

We also described a potential weakness of FCSR based pseudorandom generators, which should occur if the transition function is linear during several consecutive iterations. However, we show that such a bad behavior cannot occur if the setup method used before extracting data is well designed. The trick to use here is to clock the automaton $n + 4$ times before to output any data.

References

[1] F. Arnault and T.P. Berger. F-FCSR: design of a new class of stream ciphers. In *Fast Software Encryption - FSE 2005, Lecture Notes in Computer Science*, vol. 3557, pp. 83–97. Springer-Verlag, 2005.

[2] F. Arnault, T.P. Berger, and C. Lauradoux. Preventing weaknesses on F-FCSR in IV mode and tradeoff attack on F-FCSR-8. ECRYPT - Stream Cipher Project Report 2005/075, 2005. http://www.ecrypt.eu.org/stream/.

[3] F. Arnault, T.P. Berger, and C. Lauradoux. Update on F-FCSR stream cipher. ECRYPT - Network of Excellence in Cryptology, Call for stream Cipher Primitives - Phase 2 2006. http://www.ecrypt.eu.org/stream/.

[4] François Arnault and Thierry P. Berger. Design and properties of a new pseudorandom generator based on a filtered FCSR automaton. *IEEE Trans. Computers*, vol. 54(11), pp.1374–1383, 2005.

[5] T. Berger and F. Arnault. Design of new pseudorandom generators based on filtered FCSR automaton. In *ECRYPT Network of Excellence - SASC Workshop Record*, pages 109–120, 2004.

[6] T.P. Berger and M. Minier. Two algebraic attacks against the F-FCSRs using the IV mode. in S. Maitra, C.E. Veni Madhavan, R. Venkatesan editors, *Progress in Cryptology - INDOCRYPT 2005*, Lecture Notes in Computer Science, vol. 3797, pp. 143–154, Springer-Verlag, 2005.

[7] Mark Goresky and Andrew Klapper. Arithmetic crosscorrelations of feedback with carry shift register sequences. *IEEE Transactions on Information Theory*, vol. 43(4), pp.1342–1345, 1997.

[8] A. Klapper and M. Goresky. *Feedback shift registers, 2-adic span, and combiners with memory*, *Journal of Cryptology*, vol. 10, pp. 111–147, 1997.

[9] Mark Goresky and Andrew Klapper. Fibonacci and Galois representations of feedback-with-carry shift registers. *IEEE Transactions on Information Theory*, vol. 48(11), pp. 2826–2836, 2002.

[10] E. Jaulmes and F. Muller. Cryptanalysis of the F-FSCR stream cipher family. In *proceedings of 12th annual workshop on Selected Areas in Cryptography*, Lecture Notes in Computer Science, vol. 3897, pp. 20–35, Springer-Verlag, 2005.

[11] A. Klapper and M. Goresky. 2-adic shift registers. In *Fast Software Encryption - FSE'93, Lecture Notes in Computer Science*, vol. 809, pp. 174–178. Springer-Verlag, 1993.

[12] N. Koblitz. p-adic numbers, p-adic analysis and zeta-functions. Springer-Verlag, 1997.

A New Approach for FCSRs*

François Arnault[1], Thierry Berger[1], Cédric Lauradoux[2], Marine Minier[3], and Benjamin Pousse[1]

[1] XLIM (UMR CNRS 6172), Université de Limoges
123 avenue Albert Thomas, F-87060 Limoges Cedex - France
first name.name@xlim.fr
[2] Information Security Group
UCL / INGI / GSI
2, Place Saint Barbe
B-1348 Louvain-la-Neuve - Belgium
cedric.lauradoux@uclouvain.be
[3] Lyon University - CITI Laboratory - INSA de Lyon
6, avenue des arts, 69621 Villeurbanne Cedex - France
marine.minier@insa-lyon.fr

Abstract. The Feedback with Carry Shift Registers (FCSRs) have been proposed as an alternative to Linear Feedback Shift Registers (LFSRs) for the design of stream ciphers. FCSRs have good statistical properties and they provide a built-in non-linearity. However, two attacks have shown that the current representations of FCSRs can introduce weaknesses in the cipher. We propose a new "ring" representation of FCSRs based upon matrix definition which generalizes the Galois and Fibonacci representations. Our approach preserves the statistical properties and circumvents the weaknesses of the Fibonacci and Galois representations. Moreover, the ring representation leads to automata with a quicker diffusion characteristic and better implementation results. As an application, we describe a new version of F-FCSR stream ciphers.

Keywords: Stream cipher, FCSRs, ℓ-sequence, ring FCSRs.

1 Introduction

The FCSRs have been proposed by Klapper and Goresky [1,2,3] as an alternative to LFSRs for the design of stream ciphers. FCSRs share many of the good properties of LFSRs: sequences with known period and good statistical properties. But unlike LFSRs, they provide an intrinsic resistance to algebraic and correlation attacks because of their quadratic feedback function. However, two recent results [4,5] have shown weaknesses in stream ciphers using either the Fibonacci or Galois FCSR. Hell and Johansson [5] have exploited the bias in the carries behaviour of a Galois FCSR to mount a very powerful attack against

* This work was partially supported by the french National Agency of Research: ANR-06-SETI-013.

M.J. Jacobson Jr., V. Rijmen, and R. Safavi-Naini (Eds.): SAC 2009, LNCS 5867, pp. 433–448, 2009.

the F-FCSR stream cipher [6,7]. Fisher *et al.* [4] have considered an equivalent of the F-FCSR stream cipher based upon a Fibonacci FCSR to study the linear behavior of the induced system.

We present a new approach for FCSRs, which we call the ring representation or ring FCSR. This representation is based on the adjacency matrix of the automaton graph. A ring FCSR can be viewed as a generalization of the Fibonacci and Galois representations. Similar structure has been widely studied for the LFSR case as in [8,9,10], and is a building block of the stream cipher Pomaranch where LFSRs are used [11]. However, we present here for the first time this structure in the FCSR case.

A Fibonacci FCSR, has a single feedback function which depends on multiple inputs. A Galois FCSR has multiple feedbacks which all share one common input. A ring FCSR can be viewed as a trade-off between the two extreme cases. It has several feedback functions with different inputs. An example of ring FCSR is shown in Fig. 1.

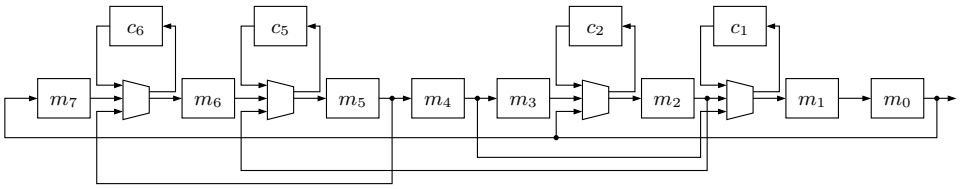


Fig. 1. An example of a ring FCSR ($q = -347$)

Ring FCSRs have many advantages, while preserving all the good and traditional properties of Galois/Fibonnacci FCSRs (known period, large entropy,...). The main one is that the attack of Hell and Johansson [5] does not work with Ring FCSR. Also, they have better diffusion properties. Moreover, ring representation allows fine tune in the implementation.

Section 2 gives an overview on FCSRs theory and classical representations. Section 3 presents ring FCSRs. We discuss implementation in Section 4 and a new version of F-FCSR is proposed in Section 5.

2 Theoretical Background

First, we will recall some basic properties of 2-adic integers. For a more theoretical approach the reader can refer to [1,2,12,13,14].

2.1 2-adic Numbers and Period

A 2-adic integer is formally a power series $s = \sum_{i=0}^{\infty} s_i 2^i$, $s_i \in \{0, 1\}$. This series always converges if we consider the 2-adic topology. The set of 2-adic integers is denoted by $\mathbb{Z}_2$. Addition and multiplication in $\mathbb{Z}_2$ can be performed by reporting the carries to the higher order terms, i.e. $2^n + 2^n = 2^{n+1}$ for all $n \in \mathbb{N}$. If there

exists an integer N such that $s_n = 0$ for all $n \geq N$, then s is a positive integer. Every odd integer q has an inverse in $\mathbb{Z}_2$.

The following property gives a complete characterization of eventually periodic binary sequences in terms of 2-adic integers (see [13] for the proof).

Property 1. *Let $S = (s_n)_{n\in\mathbb{N}}$ be a binary sequence and let $s = \sum_{i=0}^{\infty} s_i 2^i$ be the corresponding 2-adic integer. The sequence S is eventually periodic if and only if there exist two numbers p and q in $\mathbb{Z}$, q odd, such that $s = p/q$.*

Moreover, S is strictly periodic if and only if $pq \leq 0$ and $|p| \leq |q|$. In this case, we have the relation $s_n = (p \cdot 2^{-n} \mod q) \mod 2$.

The period of S is the order of 2 modulo q, i.e., the smallest integer T such that $2^T \equiv 1 \pmod{q}$. The period satisfies $T \leq |q| - 1$. If q is prime, then T divides $|q| - 1$. If $T = |q| - 1$, the sequence S is called an ℓ-sequence. As detailed in [1,2,13,15], ℓ-sequences have many proved properties that could be compared to the ones of m-sequences: known period, good statistical properties, fast generation, etc. In summary, FCSRs have almost the same properties as LFSRs but they have a nonlinear structure.

2.2 Galois FCSRs

A Galois FCSR (as shown in Fig. 2) consists of an n-bit main register $M = (m_0, \ldots, m_{n-1})$ with some fixed feedback positions $d_0, \ldots, d_{n-1}$. All the feedbacks are controlled by the cell m_0, and $n - 1$ binary carry cells $C = (c_0, \ldots, c_{n-2})$. At time t, an automaton in state (M, C) is updated in the following way:

1. Compute the sums $x_i = m_{i+1} + c_i d_i + m_0 d_i$ for all i such that $0 \leq i < n$ with $m_n = 0$ and $c_{n-1} = 0$ and where m_0 represents the feedback bit;
2. Update the state as follows: $m_i \leftarrow x_i \mod 2$ for all $i \in [0..n-1]$ and $c_i \leftarrow x_i \operatorname{div} 2$ for $0 \leq i < n$ for all $i \in [0..n-2]$.

The reader can refer to [13] for a complete description of Galois FCSRs and some properties. We recall however a very important one, found in [16].

Property 2. *Let $q = 1 - 2\sum_{i=0}^{n-1} d_i 2^i$ and $r_i = \sum_{t=0}^{\infty} m_i(t) 2^t$ (for $0 \leq i < n$); r_i is the 2-adic integer corresponding to the sequence observed in the i-th cell of the main register M. Then, for all $0 \leq i < n$, there exists $p_i \in \mathbb{Z}$ such that $r_i = p_i/q$.*

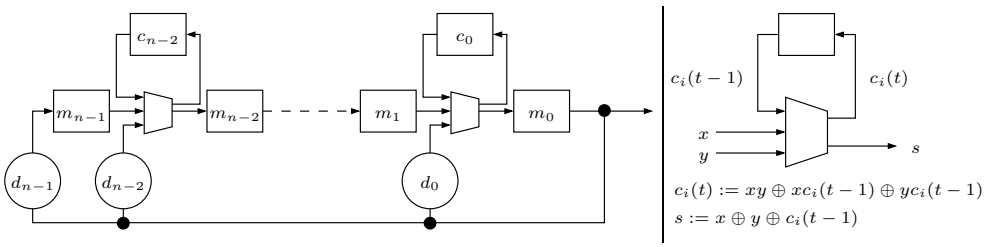


Fig. 2. A Galois FCSR and 2-bit adder with carry

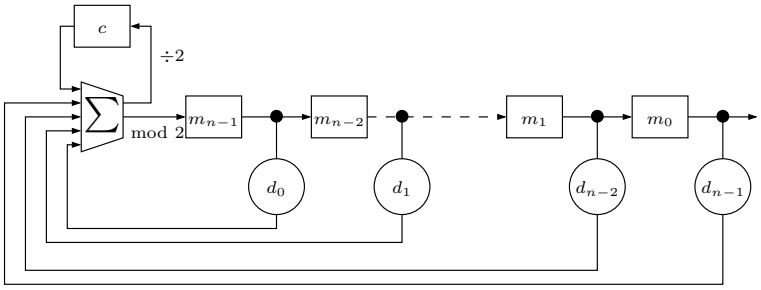


Fig. 3. A Fibonacci FCSR

In a Galois FCSR, a single cell controls all the feedbacks. As a consequence, there exist some correlations between the carries values and the feedback value. This fact is the basis of the attack presented in [5].

2.3 Fibonacci FCSRs

A Fibonacci FCSR (represented in Fig. 3) is composed of a main register $M = (m_0, \ldots, m_{n-1})$ with n binary cells. The binary feedback taps $(d_0, \ldots, d_{n-1})$ are associated to an additional carry register c of $w_H(d)$ binary cells, where $w_H(d)$ is the Hamming weight of $d = (1 + |q|)/2$.

An automaton in state (M, c) is updated in this way:

1. compute the sum $x = c + \sum_{i=0}^{n-1} m_i d_{n-1-i}$;
2. then, update the state: $M \leftarrow (m_1, \ldots, m_{n-1}, x \mod 2),\ c \leftarrow x \operatorname{div} 2$.

As shown in [13], Property 2 also holds for Fibonacci FCSRs : the sequence observed in a cell m_i is a 2-adic integer.

The cell m_{n-1} is the only one with a non-linear behaviour in a Fibonacci FCSR. As shown in [4], an attack can be carried out if a linear filter is used with a Fibonacci FCSR.

3 A New Approach for FCSRs

Galois and Fibonacci FCSRs are two different automata with similar properties, as seen in the previous section. In a Galois FCSR, the first cell m_0 modifies $w_H(d)$ cells of the main register. In a Fibonacci FCSR, the cell m_{n-1} is modified by $w_H(d)$ cells of the main register. Ring representation of FCSRs is a trade-off between these extreme cases.

Definition 1. *A ring FCSR is an automaton composed of a main shift register of n binary cells $m = (m_0, \ldots, m_{n-1})$, and a carry register of n integer cells $c = (c_0, \ldots, c_{n-1})$. It is updated using the following relations:*

$$\begin{cases} m(t+1) = Tm(t) + c(t) \mod 2 \\ c(t+1) = Tm(t) + c(t) \operatorname{div} 2 \end{cases} \tag{1}$$

$$T_R = \begin{pmatrix} 0&1&0&0&0&0&0&0\\ 0&0&1&0&1&0&0&0\\ 1&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&1&0&0\\ 0&0&1&0&0&0&1&0\\ 0&0&0&0&0&1&0&1\\ 1&0&0&0&0&0&0&0 \end{pmatrix}$$

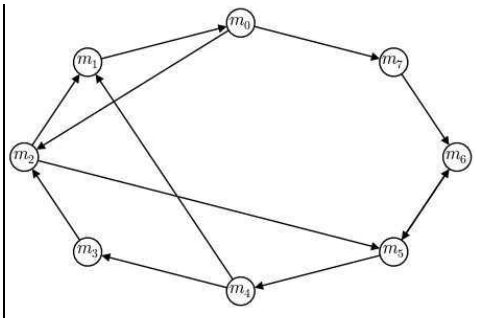


Fig. 4. Matrix and graph representation of FCSR presented in Fig.1

where T is a $n \times n$ matrix with coefficients 0 or 1 in $\mathbb{Z}$, called transition matrix, of this form:

$$\begin{pmatrix} * & 1 & & & & \\ & * & 1 & & (*) & \\ & & * & 1 & & \\ & & & \ddots & \ddots & \\ & & (*) & & * & 1 \\ 1 & & & & & * \end{pmatrix}$$

Note that $\div 2$ is the traditional expression: $X \ \mathrm{div}\, 2 = \frac{X-(X \mod 2)}{2}$.

Ring FCSRs differ from Fibonacci and Galois FCSRs in the fact that any cell can be used as a feedback for any other cell. A more convenient way to draw ring FCSRs is presented in Figure 4, which represents the same FCSR as the one in Figure 1.

3.1 Remarks on the Transition Matrix

According to Definition 1, we have the following property, where $t_{i,j}$ is the element at the i-th row and j-th column:

$$T = (t_{i,j})_{0 \le i,j < n} \text{ with } t_{i,j} = \begin{cases} 1 & \text{if cell } m_j \text{ is used to update cell } m_i, \\ 0 & \text{otherwise.} \end{cases}$$

As the main register of a ring FCSR is by definition a shift register, the over-diagonal of the transition matrix T is full of ones, i.e. for all $0 \le i < n$ we have $t_{i,i+1 \bmod n} = 1$. For example, the FCSR presented in Fig.1 has the following transition matrix T_R (and $q = -347$):

This notation agrees with the one proposed in [13]. In particular, Galois and Fibonacci FCSRs have respectively transition matrices T_G and T_F of the form:

$$T_G = \begin{pmatrix} d_0 & 1 & & & & \\ d_1 & 0 & 1 & & (0) & \\ d_2 & & 0 & 1 & & \\ \vdots & & & \ddots & \ddots & \\ d_{n-2} & & (0) & & 0 & 1 \\ 1 & & & & & 0 \end{pmatrix} \qquad T_F = \begin{pmatrix} 0 & 1 & & & & \\ & 0 & 1 & & (0) & \\ & & 0 & 1 & & \\ & & & \ddots & \ddots & \\ (0) & & & & 0 & 1 \\ 1 & d_{n-2} & \dots & d_2 & d_1 & d_0 \end{pmatrix}$$

3.2 Characterizing the Cells Content

Definition 1 introduces the transition matrix of a ring FCSR. We explain now how the value q can be computed from the transition matrix T.

Let $m_i(t)$ denote the content of the i-th cell of the main register at time t and $M_i(t)$ the series observed in this cell, from time t:

$$M_i(t) = \sum_{k \in \mathbb{N}} m_i(t+k)2^k.$$

From Equation 1, we derive the following relation

$$M(t+1) = TM(t) + c(t) \tag{2}$$

where $M(t) = (M_0(t), \cdots, M_{n-1}(t))$, and $c(t) = (c_0(t), \cdots, c_{n-1}(t))$ is the content of the carry register at time t.

The series $M_i(t)$ and the vector $M(t)$ play a fundamental role in our approach. We have the following important generalisation of Property 2.

Theorem 1. *The series $M_i(t)$ observed in the cells of the main register are 2-adic expansion of p_i/q with $p_i \in \mathbb{Z}$ and with $q = \det(I - 2T)$.*

Proof. According to the definition of $M_i(t)$ and to Definition 1, we have $M(t) = 2M(t+1) + m(t)$ where $m(t)$ is a binary vector of size n. Using Equation 2, we get:

$$(I - 2T) \cdot M(t) - 2 \cdot c(t) - m(t) = 0.$$

Considering the adjugate of $I - 2T$, we obtain:

$$\det(I - 2T) \cdot M(t) = Adj(I - 2T)(m(t) + 2 \cdot c(t)).$$

In this relation, the right member is a vector of integers $(p_0(t), \ldots, p_{n-1}(t))$. Dividing by $\det(I - 2T)$, we obtain $M_i(t) = p_i(t)/\det(I - 2T)$.

Lemma 1. *With the notation of Theorem 1, if $q = \det(I - 2T)$ is prime, and if the order of 2 in $\mathbb{Z}/q\mathbb{Z}$ is maximal, then each M_i is an ℓ-sequence.*

4 Implementation Properties

We detail in this section the new implementation characteristics of ring FCSRs. All this section applies also to LFSRs by replacing addition with carry with addition modulo 2.

Path/fan-out – The Galois FCSR is considered in many works [13,17,18] as the best representation for hardware implementation. It has a better critical path, i.e, a shorter longest path, than a Fibonacci FCSR. A drawback of the Galois representation is that the fan-out of the feedback cell m_0 is $w_H(d)$ with $d = (1 + |q|)/2$. At the opposite, the Fibonacci representation has a fan-out of 2. A ring FCSR allows the designer to tune both the critical path and the fan-out through the choice of the transition matrix:

Table 1. Comparison of the different representations

	Fibonacci	Galois	Ring
Path	$\lceil \log_2(w_H(d)) \rceil$	1	$\max(\lceil \log_2(w_H(a_i)) \rceil)$
Fan-out	2	$w_H(d)$	$\max(w_H(b_i))$
Cost ($\#_{adders}$)	$w_H(d)-1$	$w_H(d)-1$	$w_H(T)-n$

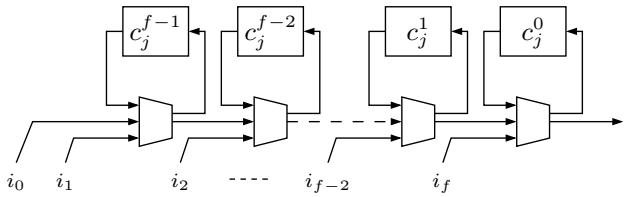


Fig. 5. A naive adder

- the critical path is given by the row a_i with the largest number of 1s;
- the fan-out is given by the column b_i with the largest number of 1s.

We compare in Table 1 the critical path, the fan-out and the cost of the different representations of an FCSR. We have expressed the critical path as the number of adders crossed. The choice of the adder has also an impact on the path of a ring FCSR. A naive adder (Fig. 5) composed of a serialisation of generic adder leads to a path of $max(w_H(a_i))-1$ adders. However, it is possible to exploit the commutativity to perform additions in parallel. This reduces the critical path to $max(\lceil \log_2(w_H(a_i)) \rceil)$ adders.

For each given q, it should be possible to find a transition matrix corresponding to a critical path with only one adder and a fan-out equal to 2. This is the case of the ring FCSR given in Fig. 1.

Cost – Ring FCSR have implementations which require fewer gates than Fibonacci/Galois equivalent ones. This possibility was first observed in [10] for LFSRs. However, the solution proposed in [10] is specific to LFSRs and cannot be applied systematically to FCSRs. The number $\#_{adders}$ of 2-bit adders required in the different representations of an n-bit FCSR is shown in Table 1. Ring representation is the only one that allows to find an implementation with less than $(w_H(d)-1)$ 2-bit adders. For $q=-347$, a Galois or Fibonacci representation leads to $\#_{adders}=4$. A ring representation with the following transition matrix T_R:

$$T_R = \begin{pmatrix} 0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&0&1&0&0&0&1 \\ 0&1&0&0&1&0&0&0 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&1 \\ 1&0&1&0&0&0&0&0 \end{pmatrix}$$

leads to an implementation with $\#_{adders} = 3$, a fan-out of 2 and a critical path of 1 adder.

Side-channel attacks – It seems possible to work out an equivalent of the side-channel attack of Joux and Delaunay [18] on Galois FCSR using the results of Hell and Johansson [5]. Such an attack would exploit the power consumption to recover the feedback m_0 (because of the excessive fan-out of the feedback cell) and therefore how the carry cells are modified. As the ring FCSR has a reduced fan-out and uncorrelated carries, it is a better alternative to prevent side-channel attacks.

5 F-FCSR Based on Ring Representation

In this section, we propose a generic algorithm to construct F-FCSR stream ciphers based upon a ring FCSR with a linear filter. We give two particular examples which are F-FCSR-H v3 and F-FCSR-16 v3. Any designer using the proposed algorithm could generate its own stream cipher according to the following parameters:

- key length k and IV length v that will provide the corresponding size $n := k + v$ of the T matrix (usually $k = v$);
- the number u of bits output at each clock taken between 1 and $n/16$ to ensure a hard inversibility of the filter. Moreover for later design we require u to be a divisor of n;
- the number of willing feedbacks ℓ usually taken between $n/2-5$ and $n/2+5$ to ensure a sufficient non linear structure and a sufficiently weighted filter.

The algorithm is composed of 3 particular steps: the choice of the matrix T, the choice of the linear filter and the key/IV setup.

5.1 The Choice of the Matrix T

According to the remarks in Section 3, we pick a $n \times n$ random matrix T with the following requirements:

- the matrix must be composed of 0 and 1 and with a general weight equal to $n + \ell$;
- the over-diagonal must be full of 1 and $t_{n-1,0} = 1$ (to preserve the ring structure of the automaton);
- the number of ones for a given row or a given column must be at most two. This last condition allows a better diffusion by maximizing the number of cells reached by the feedbacks. It also provides uncorrelated carries and a fan-out bounded by 2.

For each picked matrix with the previous requirements, test if:

1. $\log_2(q) \geq n$; $\det(T) \neq 0$;
2. $q = \det(I - 2T)$ is prime; the order of 2 modulo q is $|q| - 1$.

The first condition ensures a non-degenerated matrix. The second ensures good statistical properties and a long period.

This matrix completely defines the ring FCSR. The diffusion speed (which is faster than in Galois/Fibonacci FCSRs) is related to the diameter d of the transition graph. This diameter is the maximal distance between two cells of the main register. In other words, d is the distance after which any cell of the main register have been influenced by any other cell through the feedbacks. It corresponds to the minimal number of clocks required to have all the cells of the main register influenced by any other cell. d should be small for better diffusion.

5.2 The Filter

As in the previous versions of F-FCSR [7], we use a linear filter to extract the keystream in order to break the 2-adic structure of the automaton. This also prevents linearization attacks over the set of 2-adic numbers. The filter includes the cells of the main register which receive a feedback to prevent correlation attacks. The periodic structure of the filter in the previous versions of F-FCSR has been exploited in [5] to speed up the linear part of the attack. We prefer now a non periodic structure:

- let $F = \{m_{f_0}, \cdots, m_{f_{\ell-1}}\}$ be the set of all the cells m_i that receive a feedback and indexed in this way: the row f_i of the matrix T has more than one 1 for $0 \leq i < \ell$, and $f_i < f_{i+1}$.
- The u bits of output are: $\forall\, 0 \leq i < u,\ z_i = \bigoplus_{j \equiv i \mod u} m_{f_j}$.

5.3 Key and IV Setup

As shown in [5], if at a given time, the FCSR is in a synchronized state (i.e. a state from which after a finite number of steps the automaton will return, i.e. a state belonging to the main cycle), adjacent states of the main cycle could be directly deduced using only multiplications over $\mathbb{Z}/q\mathbb{Z}$. Moreover, as shown in [16], a Galois FCSR is synchronized in at most $n + 4$ clocks, but in reality, few clocks are sufficient. So, to completely avoid the weakness of the key and IV setup used in [5], we prefer to maintain a non synchronized state during the key and IV setup. The new key and IV setup creates a transformation that is really hard to invert, in order to prevent a direct key recovery attack.

However, using a ring FCSR leads to a new problem: we can not ensure the entropy of the automaton. In the case of F-FCSR with Galois or Fibonacci structure, zeroing the content of the carry register prevents collisions (i.e. one point of the states graph with two preimages) and warrants a constant entropy. This particular property comes from the particular structure of the adjoint matrix $(I - 2T)^*$, which has successive powers of two in its first row in case of a Galois FSCR (in a Fibonacci FCSR, a similar property holds for the last row). In the ring case, no obvious structure exists in $(I - 2T)^*$. Note that in this case the collisions search becomes an instance of the subset sum problem, with a complexity equals to $2^{n/2}$ (if the carries are zeroes) or $2^{3n/2}$ (in the general case).

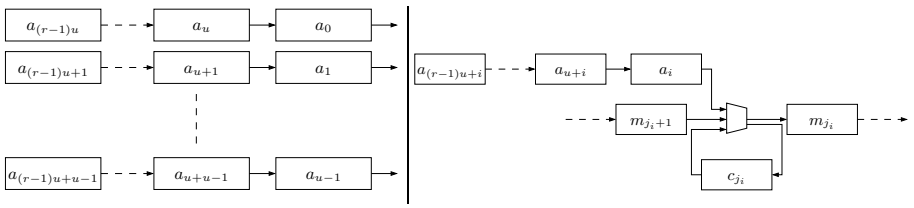


Fig. 6. Disposition of the cells $a_0, \ldots, a_{n-1}$ in u shift registers and connection of a shift register in position j_i

Thus, the new key and IV setup aims to stay on non-synchronized states as long as possible and to limit the entropy loss. We connect at u different places shift registers of length $r = n/u$ (this corresponds to adding n binary cells $a_0, \ldots, a_{n-1}$ at different places as shown in Fig. 6).

The u positions denoted by $J := \{j_0 < \cdots < j_{u-1}\}$ where the u shift registers are connected have been chosen such that, for all $0 \leq i < u$, no adder exists between cells m_{j_i+1} and m_{j_i} (i.e. $w_H(R_{j_i}) = 1$ where R_{j_i} is the j_i^{th} row of the matrix T). Each shift register is connected using a 2-bit adder with carry (as shown in Fig. 6). The content of cell m_{j_i} after transition depends on the values of m_{j_i+1}, a_i and of the carry cell c_{j_i}.

With these u shift registers inserted in the ring FCSR, the key and IV setup works as follows:

- Initialize $(a_0, \ldots, a_{n-1})$ with $(K \| 0^{n-k-v} \| IV)$, $M \leftarrow 0$, $C \leftarrow 0$.
- The FCSR is clocked r times. At each clock, the FCSR is filtered using F to produce a u bits vector $z_0, \ldots, z_{u-1}$ used to fill back $a_{(r-1)u}, \ldots, a_{(r-1)u+u-1}$: $a_{(r-1)u+i} \leftarrow z_i$ for $0 \leq i < u$.
- The FCSR is clocked $\max(r, d+4)$ times discarding the output.

The first step of the key and IV setup allows an initial diffusion of the key through the simple shift registers. The next r clocks helps a complete diffusion of the IV and of the key in the FCSR. The diffusion is complete at the end of the key and IV setup. If an attacker is able to recover the state just at the end of the key and IV setup, he won't be able to use this information to recover the key because of the occurence of non-synchronized states that are hard to inverse: for a given m_{k+1} bit value of the main register, the values c_k and m_k producing m_{k+1} are not unique and this leads to a combinatorial explosion when an attacker wants to recover a previous state.

As previously mentioned, this construction does not provide a bijection and behaves more like a random function. From this point, two attacks are essentially possible: direct collisions search and time memory data trade-off attack for collisions search built upon entropy loss. As mentioned before, direct collisions search has a cost of $2^{(n/2)}$ if the attacker is able to clear the carry bits. With the use of a ring FCSR that does not allow a direct control of the carry bits through the feedback bit, the probability to force to 0 the carry bits is about $2^{-\ell}$. Thus such an attack is more expensive than a key exhaustive search. In the other cases, the corresponding complexity is $2^{(3n/2)}$ preventing collisions search.

TMDTO attacks are possible if a sufficient quantity of entropy is lost. As studied in [19], considering that the key and IV setup are random function, the induced entropy loss is about 1 bit, so considering an initial entropy equal to n bits, the entropy after the key and IV setup is close to $n-1$ bits. Is it possible to exploit this entropy loss for a collisions search in a TMDTO attack? A well-known study case is the attack proposed in [20] by J. Hong and W.H. Kim against the stream cipher MICKEY. Even if this attack seems to work, A. Rock has shown in [19] that the query complexity in the initial states space could not be significantly reduced and that the attacks based on the problem of entropy loss are less efficient than expected especially regarding the query complexity. So, we conjecture, that our key and IV setup behaves as a random function, and that the induced entropy loss is not sufficient to mount a complete TMDTO attack for collisions search taking into account the query complexity.

5.4 F-FCSR-H v3 and F-FCSR-16 v3

The details of the two constructions, especially the corresponding T matrices, are given respectively in Appendix A and B. The respective parameters are the following ones:

- For F-FCSR-H v3: $k = 80$, $v = 80$, $\ell = 82$, $n = 160$, $u = 8$, $d = 24$;
- For F-FCSR-16 v3: $k = 128$, $v = 128$, $\ell = 130$, $n = 256$, $u = 16$, $d = 28$.

These two automata have been chosen with an additional property: $(|q| - 1)/2$ prime. This condition ensures maximal period for the output stream. However this condition is hard to fill. So we don't require this condition in the general case.

5.5 Resistance against Known Attacks

We do not discuss here resistance against traditional attacks such as correlation / fast correlation attacks, guess and determine attacks, algebraic attacks, etc. Some details about this can be found in [7]. Resistance against TMDTO attacks was considered in Section 5.3. We focus now on the two recent attacks [5] and [4] against FCSR and F-FCSR.

The attack presented in [5] against F-FCSR, which is based on a Galois FCSR, relies on the existence of correlations between the carries and the feedback values. More precisely, the control of the m_0 bit leads to the control of the feedback values. If the feedback can be forced to 0 during t consecutive clocks, the behavior of the stream cipher becomes linear, and its synthesis is possible by solving a really simple system. This linear behavior happens with a probability about 2^{-t} for a Galois FCSR. If instead a ring FCSR is used, this probability decreases to $2^{-t \cdot k}$ where k is the number of cells of the main register controlling a feedback. Thus, for k values corresponding to most ring FCSR, the linear behavior probability becomes so small that the cost of the corresponding attack becomes higher than an exhaustive search. Also the attack from [5] relies on situations where the carries remain constant during t consecutive clocks. We made an experiment

with F-FCSR-H v3 to search for states for which carries does not change during transition. Looking over 2^{38} states, we found only 41 different states for which carries remains constant after one transition. We found none for which carries remains constant after two transitions.

In [4], the authors propose a linearization attack against a linearly filtered Fibonacci FCSR. This attack does not affect any version of F-FCSR. In a Fibonacci FCSR, the carries only influence one bit of the main register at each clock. Thus, if one could imagine to build a F-FCSR using a Fibonacci FCSR, such a generator would be subject to an attack where the control of the carries leads to the control of a part of the main register. Thus, we recommend to NOT use a Fibonacci FCSR in a linearly filtered stream cipher.

6 Conclusion and Future Work

In this paper, we have presented a new approach for FCSRs that unifies the two classical representations. We can obtain, with the ring representation, better diffusion characteristics and faster implementations. The recent attacks designed against F-FCSR are prevented, when using a ring FCSR, as shown in Section 5.

References

1. Klapper, A., Goresky, M.: 2-adic shift registers. In: Anderson, R. (ed.) FSE 1993. LNCS, vol. 809, pp. 174–178. Springer, Heidelberg (1994)
2. Klapper, A., Goresky, M.: Feedback shift registers, 2-adic span and combiners with memory. Journal of Cryptology 10(2), 111–147 (1997)
3. Klapper, A.: A survey of feedback with carry shift registers. In: Helleseth, T., Sarwate, D., Song, H.-Y., Yang, K. (eds.) SETA 2004. LNCS, vol. 3486, pp. 56–71. Springer, Heidelberg (2005)
4. Fischer, S., Meier, W., Stegemann, D.: Equivalent Representations of the F-FCSR Keystream Generator. In: ECRYPT Network of Excellence - SASC Workshop, pp. 87–94 (2008), `http://www.ecrypt.eu.org/stvl/sasc2008/`
5. Hell, M., Johansson, T.: Breaking the F-FCSR-H stream cipher in real time. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 557–569. Springer, Heidelberg (2008)
6. Arnault, F., Berger, T.P.: F-FCSR: Design of a new class of stream ciphers. In: Gilbert, H., Handschuh, H. (eds.) FSE 2005. LNCS, vol. 3557, pp. 83–97. Springer, Heidelberg (2005)
7. Arnault, F., Berger, T.P., Lauradoux, C.: Update on F-FCSR Stream Cipher. ECRYPT - Network of Excellence in Cryptology (Call for stream Cipher Primitives - Phase 2 2006) (2006), `http://www.ecrypt.eu.org/stream/`
8. Roggeman, Y.: Varying feedback shift registers. In: Quisquater, J.-J., Vandewalle, J. (eds.) EUROCRYPT 1989. LNCS, vol. 434, pp. 670–679. Springer, Heidelberg (1990)
9. Jansen, C.J., Helleseth, T., Kholosha, A.: Cascade jump controlled sequence generator and pomaranch stream cipher (version 2). eSTREAM, ECRYPT Stream Cipher Project, Report 2006/006 (2006), `http://www.ecrypt.eu.org/stream`
10. Mrugalski, G., Rajski, J., Tyszer, J.: Ring generators - new devices for embedded test applications. IEEE Trans. on CAD of Integrated Circuits and Systems 23(9), 1306–1320 (2004)

11. Jansen, C.J., Helleseth, T., Kholosha, A.: Pomaranch version 3. eSTREAM, ECRYPT Stream Cipher Project (2006), http://www.ecrypt.eu.org/stream
12. Koblitz, N.: p-adic numbers, p-adic analysis and Zeta-Functions. Springer, Heidelberg (1997)
13. Goresky, M., Klapper, A.: Fibonacci and Galois representations of feedback-with-carry shift registers. IEEE Transactions on Information Theory 48(11), 2826–2836 (2002)
14. Arnault, F., Berger, T.P.: Design and Properties of a New Pseudorandom Generator Based on a Filtered FCSR Automaton. IEEE Transaction on Computers 54(11), 1374–1383 (2005)
15. Lauradoux, C., Röck, A.: Parallel generation of ℓ-sequences. In: Golomb, S.W., Parker, M.G., Pott, A., Winterhof, A. (eds.) SETA 2008. LNCS, vol. 5203, pp. 299–312. Springer, Heidelberg (2008)
16. Arnault, F., Berger, T.P., Minier, M.: Some Results on FCSR Automata With Applications to the Security of FCSR-Based Pseudorandom Generators. IEEE Transactions on Information Theory 54(2), 836–840 (2008)
17. Goldberg, I., Wagner, D.: Architectural considerations for cryptanalytic hardware. Technical report, Secrets of Encryption Research, Wiretap Politics & Chip Design (1996)
18. Joux, A., Delaunay, P.: Galois LFSR, embedded devices and side channel weaknesses. In: Barua, R., Lange, T. (eds.) INDOCRYPT 2006. LNCS, vol. 4329, pp. 436–451. Springer, Heidelberg (2006)
19. Röck, A.: Stream ciphers using a random update function: Study of the entropy of the inner state. In: Vaudenay, S. (ed.) AFRICACRYPT 2008. LNCS, vol. 5023, pp. 258–275. Springer, Heidelberg (2008)
20. Hong, J., Kim, W.H.: TMD-Tradeoff and State Entropy Loss Considerations of Streamcipher MICKEY. In: Maitra, S., Veni Madhavan, C.E., Venkatesan, R. (eds.) INDOCRYPT 2005. LNCS, vol. 3797, pp. 169–182. Springer, Heidelberg (2005)

A Description of the Transition Matrix for F-FCSR-H v3

Input parameters: $k = 80$ (key length), $v = 80$ (IV length), $\ell = 82$ (number of feedbacks), $n = 160$ (size of T), $u = 8$ (number of output bits), $d = 24$ (diameter of the graph).

We give here the description of the transition matrix $T = (t_{i,j})_{0 \le i,j < 160}$ (see Fig. 7 for graphic representations):

- For all $0 \le i < 160$, $t_{i,i+1 \mod 160} = 1$;
- For all $(i, j) \in S$, $t_{i,j} = 1$, where $S = \{$ (1, 121); (2, 133); (4, 44); (5, 82); (9, 38); (11, 40); (12, 54); (14, 105); (15, 42); (16, 63); (18, 80); (19, 136); (20, 2); (21, 35); (23, 28); (25, 137); (28, 131); (31, 102); (36, 41); (39, 138); (40, 31); (42, 126); (44, 127); (45, 77); (46, 110); (47, 86); (48, 93); (49, 45); (51, 17); (54, 8); (56, 7); (57, 150); (59, 25); (62, 51); (63, 129); (65, 130); (67, 122); (73, 148); (75, 18); (77, 46); (79, 26); (80, 117); (81, 1); (84, 72); (86, 60); (89, 15); (90, 89); (91, 73); (93, 12); (94, 84); (102, 141); (104, 142); (107, 71); (108, 152); (112, 92); (113, 83); (115, 23); (116, 32); (118, 50); (119, 43); (121, 34); (124, 13); (125, 74); (127, 149); (128, 90); (129, 57); (130, 103); (131, 134); (132, 155); (134, 98); (139, 24);

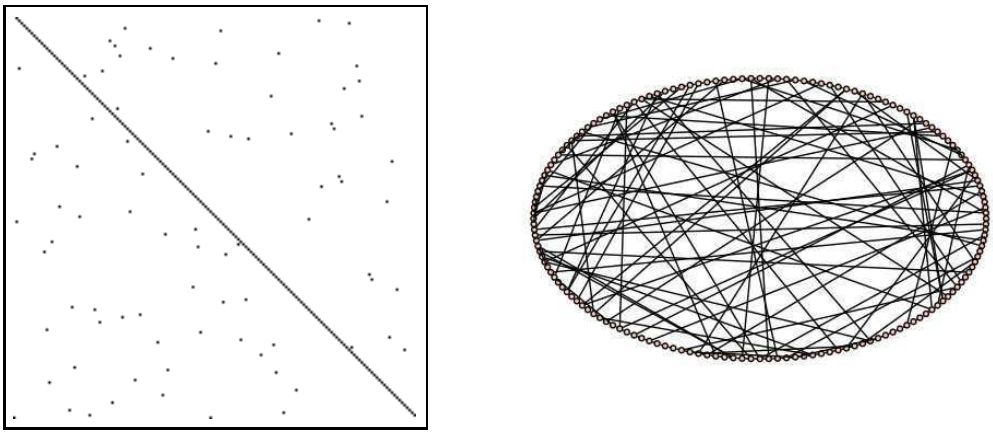


Fig. 7. Matrix representation and graph representation of the matrix T chosen for F-FCSR-H v3

(140, 61); (141, 104); (144, 48); (145, 14); (148, 112); (150, 59); (153, 39); (156, 22); (157, 107); (158, 30); (159, 78) $\}$;
- Otherwise, $t_{i,j} = 0$.

- The corresponding q value is (in decimal notation):

$$q = 174161873672323786281235399625569968955252645088 3$$

- The set J (for the first part of the Key/IV setup) is:

$$J = \{3, 22, 43, 64, 83, 103, 123, 143\}$$

- The 8 subfilters $F_0, \cdots, F_7$ are given by:

$$\begin{aligned}
F_0&=\{1, 15, 28, 46, 59, 79, 93, 115, 128, 141, 158\}\\
F_1&=\{2, 16, 31, 47, 62, 80, 94, 116, 129, 144, 159\}\\
F_2&=\{4, 18, 36, 48, 63, 81, 102, 118, 130, 145\}\\
F_3&=\{5, 19, 39, 49, 65, 84, 104, 119, 131, 148\}\\
F_4&=\{9, 20, 40, 51, 67, 86, 107, 121, 132, 150\}\\
F_5&=\{11, 21, 42, 54, 73, 89, 108, 124, 134, 153\}\\
F_6&=\{12, 23, 44, 56, 75, 90, 112, 125, 139, 156\}\\
F_7&=\{14, 25, 45, 57, 77, 91, 113, 127, 140, 157\}
\end{aligned}$$

B Description of the Transition Matrix for F-FCSR-16 v3

Input parameters: $k = 128$, $v = 128$, $\ell = 130$, $n = 256$, $u = 16$, $d = 28$.

We give here a description of the transition matrix $T = (t_{i,j})_{0 \le i,j < 256}$ (see Fig. 8 for graphic representations):

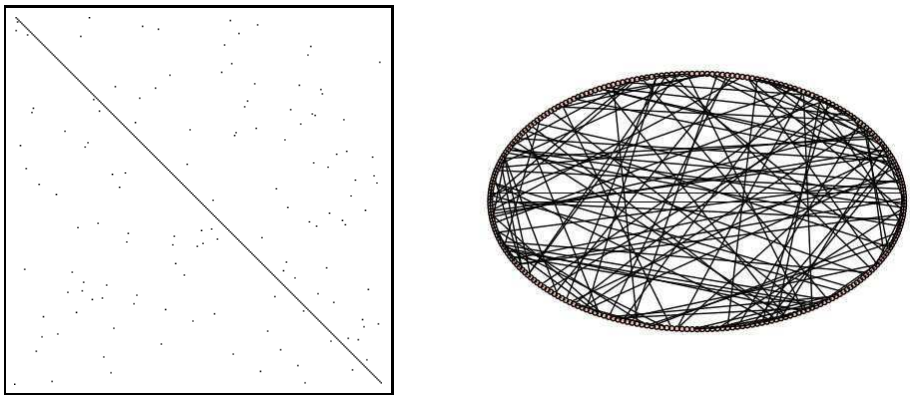


Fig. 8. Matrix representation and graph representation of the matrix T chosen for F-FCSR-16 v3

- For all $0 \leq i < 256$, $t_{i,i+1 \mod 256} = 1$;
- For all $(i, j) \in S$, $t_{i,j} = 1$, where $S = \{$ (0, 52); (2, 150); (3, 2); (5, 169); (6, 89); (8, 100); (9, 1); (11, 156); (12, 9); (13, 46); (19, 146); (20, 206); (26, 204); (31, 254); (32, 151); (38, 144); (40, 108); (46, 167); (47, 198); (48, 70); (49, 98); (50, 213); (53, 214); (56, 87); (57, 55); (58, 162); (62, 160); (63, 13); (64, 192); (65, 59); (66, 12); (67, 207); (68, 209); (71, 229); (73, 84); (74, 199); (77, 168); (78, 122); (79, 35); (80, 154); (82, 153); (85, 188); (87, 51); (89, 4); (90, 49); (93, 231); (95, 224); (97, 249); (101, 208); (102, 120); (104, 218); (105, 8); (108, 77); (109, 68); (110, 250); (113, 237); (115, 252); (116, 17); (118, 73); (119, 182); (123, 29); (124, 234); (127, 138); (132, 190); (134, 244); (136, 219); (141, 228); (142, 205); (143, 58); (144, 230); (145, 210); (146, 44); (147, 137); (148, 130); (150, 79); (152, 111); (153, 172); (154, 141); (156, 78); (157, 131); (158, 110); (159, 127); (170, 189); (171, 112); (174, 217); (175, 7); (176, 187); (177, 40); (179, 118); (181, 195); (184, 48); (186, 64); (189, 246); (190, 47); (191, 37); (192, 211); (193, 85); (194, 181); (195, 61); (196, 54); (198, 222); (199, 83); (203, 105); (204, 201); (205, 43); (206, 139); (208, 20); (210, 242); (211, 124); (213, 253); (215, 243); (216, 69); (218, 176); (220, 30); (222, 19); (223, 232); (224, 239); (225, 220); (227, 102); (231, 185); (232, 15); (234, 152); (236, 62); (238, 245); (242, 197); (245, 235); (246, 171); (247, 67); (253, 26); (254, 202) $\}$;
- Otherwise, $t_{i,j} = 0$.

- The corresponding q value is (in hexadecimal notation):

$$q = (B085834B6BFAE1541C54F7D84F42084C \\ B0568496DDD0FEA5E99AA79C022023241)$$

- The set J (for the first part of the Key/IV setup) is:

$$J = \{10, 27, 43, 59, 75, 91, 107, 122, 139, 155, 172, 187, 202, 219, 235, 251\}$$

– The 16 subfilters $F_0, \cdots, F_{15}$ are given by:

$$
\begin{aligned}
F_0&=\{0, 40, 68, 101, 134, 158, 193, 218, 253\}\\
F_1&=\{2, 46, 71, 102, 136, 159, 194, 220, 254\}\\
F_2&=\{3, 47, 73, 104, 141, 170, 195, 222\}\\
F_3&=\{5, 48, 74, 105, 142, 171, 196, 223\}\\
F_4&=\{6, 49, 77, 108, 143, 174, 198, 224\}\\
F_5&=\{8, 50, 78, 109, 144, 175, 199, 225\}\\
F_6&=\{9, 53, 79, 110, 145, 176, 203, 227\}\\
F_7&=\{11, 56, 80, 113, 146, 177, 204, 231\}\\
F_8&=\{12, 57, 82, 115, 147, 179, 205, 232\}\\
F_9&=\{13, 58, 85, 116, 148, 181, 206, 234\}\\
F_{10}&=\{19, 62, 87, 118, 150, 184, 208, 236\}\\
F_{11}&=\{20, 63, 89, 119, 152, 186, 210, 238\}\\
F_{12}&=\{26, 64, 90, 123, 153, 189, 211, 242\}\\
F_{13}&=\{31, 65, 93, 124, 154, 190, 213, 245\}\\
F_{14}&=\{32, 66, 95, 127, 156, 191, 215, 246\}\\
F_{15}&=\{38, 67, 97, 132, 157, 192, 216, 247\}
\end{aligned}
$$

Revisiting LFSRs for cryptographic applications

François Arnault, Thierry Berger, Marine Minier and Benjamin Pousse

***Abstract*—Linear Finite State Machines (LFSMs) are particular primitives widely used in information theory, coding theory and cryptography. Among those linear automata, a particular case of study is Linear Feedback Shift Registers (LFSRs) used in many cryptographic applications such as design of stream ciphers or pseudo-random generation. LFSRs could be seen as particular LFSMs without inputs.**

In this paper, we first recall the description of LFSMs using traditional matrices representation. Then, we introduce a new matrices representation with polynomial fractional coefficients. This new representation leads to sparse representations and implementations. As direct applications, we focus our work on the Windmill generators case, used for example in the E0 stream cipher and on other general applications that use this new representation.

In a second part, a new design criterion called diffusion delay for LFSRs is introduced and well compared with existing related notions. This criterion represents the diffusion capacity of an LFSR. Thus, using the matrices representation, we present a new algorithm to randomly pick LFSRs with good properties (including the new one) and sparse descriptions dedicated to hardware and software designs. We present some examples of LFSRs generated using our algorithm to show the relevance of our approach.

***Index Terms*—LFSM, LFSR, m-sequences.**

I. Introduction

Linear Finite State Machines (LFSMs) are a building block of many information theory based applications such as synchronization codes, masking or scrambling codes. They are also used for white noise signals in communication systems, signal sets in CDMA (Code Division Multiple Access) communications, key stream generators in stream ciphers, random number generators in many cryptographic primitive algorithms, and as testing vectors in hardware design.

An LFSM is a linear automaton composed of memories defined over a particular finite set $\mathcal{A}$ (typically a finite field) and where the only operation updating cells is the addition [1], [2], [3]. At each clock, it inputs n elements of $\mathcal{A}$ and outputs at least one element computed using its current state and a linear updating function based on additions. Two main classes of LFSMs could be defined: autonomous (without inputs in the updating process) and non-autonomous. This paper first recalls the traditional representation using transition matrices which is classically used to characterize autonomous and non-autonomous LFSMs. Then, it introduces a new fractional representation using rational powers series, i.e. the series are the quotient of two polynomials. Our new model is called Rational Linear Finite State Machines (RLFSMs) and is a generalization of the previous matrices representations. We present the link between the two approaches. As a particular case of study of our new representation, we focus on windmill generators defined by Smeets and Chambers in [4]. These generators are based upon particular polynomials producing in parallel v subsequences of a given LFSR sequence. Four windmill generators are used as parallel updating functions in the stream cipher E0 [5]. The windmill constructions have been first extended in [6]. In this paper, we show how we could, using the new rational representation, give a simple expression of those particular constructions and how this new theoretical representation could lead to clearly simplify the usual representation of circuits with multiple outputs at each iteration or parallelized versions of LFSRs.

In a second step, we introduce a new criterion for LFSMs to measure what we call diffusion delay. We compare this new criterion with the existing notions of auto-, cross- and simple correlations and show how our criterion captures an intrinsic behavior of the automaton. LFSMs are popular automata in many cryptographic applications and are particularly used as updating functions of stream ciphers and pseudo-random generators. Their large popularity is due to their very simple design efficient both in hardware and software and to the proved properties of the generated sequence (statistical properties, good periods,...) if the associated polynomial is primitive. In many cryptographic applications, the diffusion delay of LFSMs is not considered most of the time. In this paper, we focus on this criterion, show its link with correlation and its effectiveness for several types of automata such as FCSRs or NLFSRs. We also give a new algorithm to construct hardware and/or software efficient LFSMs with good diffusion delay called Ring LFSRs. For the hardware case, we show theoretical bounds on the number of gates required to implement a ring LFSR compared with the traditional Galois and Fibonacci LFSRs and we compare the associated traditional properties. For the software case, we compare the properties and the performances of our Ring LFSR with the LFSR involved in the stream ciphers SNOW v2.0 [7], finalist of the NESSIE project [8].

This paper is organized as follows: after giving notations, Section II gives some background about Finite State Machines (FSMs) and introduces notations. Section III presents previous works on LFSMs. Section IV introduces the new rational representation for LFSMs, detailing some examples of Windmill generators and of general applications. Section V presents the new diffusion delay criterion, shows why this criterion captures new notions and proposes hardware and software oriented implementations with respect to this criterion. Finally, Section VI concludes this paper.

This work was partially supported by the French National Agency of Research: ANR-06-SETI-013.

François Arnault, Thierry Berger and Benjamin Pousse are with XLIM (UMR CNRS 6172), Université de Limoges, 123 avenue A. Thomas, 87060 Limoges Cedex, France, firstname.name@xlim.fr

Marine Minier is with Université de Lyon, INRIA - INSA-Lyon, CITI, F-69621, Villeurbanne, France, marine.minier@insa-lyon.fr

A. Notations

The finite field with cardinal q is denoted $\mathbb{F}_q$. We denote $\mathbb{F}_q[X]$ the ring of polynomials and $\mathbb{F}_q[[X]]$ the ring of power series, both over $\mathbb{F}_q$. We also use in Sections IV and followings, the ring $\mathcal{Q}$ of rational power series, that is the ring of power series which can be written $P(X)/Q(X)$ where $P, Q \in \mathbb{F}_q[X]$ with $Q(0) \neq 0$. We recall in Theorem 2.1 that $\mathcal{Q}$ is the ring of power series that correspond to eventually periodic sequences.

We also use the notation $\mathcal{M}_{r,s}(\mathcal{R})$ for the ring of matrices with r rows and s columns over a ring $\mathcal{R}$. For convenience and not to make notations too heavy, we often write vectors v as rows $v = (v_1, \ldots, v_n)$ but also use them as column vectors in expressions such as Av where A is a matrix. Of course the correct form should be with explicit transposition as in $A\,{}^t v$ but we expect the reader not to be confused with this abuse of notation.

In Section V, we use the notation w_H for the Hamming weight. For example, the Hamming weight of a matrix is its number of nonzero entries. The Hamming weight of a polynomial is its number of non null coefficients.

II. Background

A. Linear recurring sequences

Binary sequences being often used in pseudo-random generation, this paper deals with the two elements field $\mathbb{F}_2$. However most of the results presented here have a straightforward generalization when using another finite field as base field.

Recall that a sequence $s = (s_i)_{i\in\mathbb{N}}$ over $\mathbb{F}_2$ is a *linear recurring sequence* if there exists $q_1, \ldots, q_d \in \mathbb{F}_2$ such that $s_n = q_1 s_{n-1} + \cdots + q_d s_{n-d}$ for all $n \geq d$. A binary sequence $(s_i)_{i\in\mathbb{N}}$ can be seen as a power series $s(X) = \sum_{i=0}^{\infty} s_i X^i$. In terms of power series, we have the following Theorem [1]:

Theorem 2.1: Let $s = (s_i)_{i\in\mathbb{N}}$ be a sequence over $\mathbb{F}_2$. The following statements are equivalent:

- The sequence s is a linear recurring sequence.
- The sequence s is eventually periodic, i.e. there exists $N \in \mathbb{N}$ such that $(s_i)_{i\geq N}$ is periodic.
- There exist polynomials $f(X), g(X) \in \mathbb{F}_2[X]$ with $g(0) = 1$ such that the power series $f(X)/g(X)$ is equal to $\sum_{i\in\mathbb{N}} s_i X^i$, i.e. $s(X)$ is in $\mathcal{Q}$.

Moreover, s is periodic if and only if $f(X)$ and $g(X)$ are such that $\deg f < \deg g$.

According to the Theorem 2.1 a correspondence can be built between rational power series and sequences. The period of a linear recurring sequence is determined by the polynomial $g(X)$ as shown by the following theorem [1]:

Theorem 2.2: Let $s(X) = f(X)/g(X)$ be a rational power series, with $\gcd(f(x), g(x)) = 1$. We denote by s the sequence of coefficients of $s(X)$.

- The period of s is equal to the order of X in $\mathbb{F}_2[X]/(g(X))$.
- If $g(X)$ is primitive then there exists $N \in \mathbb{N}$ such that $\sum_{i\geq N} s_i X^{i-N} = 1/g(X)$.

When the polynomial $g(X)$ is primitive, the sequence s has period $2^{\deg g} - 1$ and is called an m-sequence.

B. Adjunct matrix

Let $M = (m_{i,j})_{1\leq i,j\leq n}$ be a square matrix over a ring $\mathcal{R}$. The (i,j)-th cofactor $c_{i,j}$ of M is $(-1)^{i+j}$ times the determinant of the matrix obtained by removing the line i and the column j in M. The transpose of the cofactor matrix $(c_{i,j})$ is called the *adjunct matrix* of M and we denote it by $\mathrm{adj}(M)$. The adjunct of M has its coefficients in $\mathcal{R}$ and satisfies the following identity

$$\mathrm{adj}(M)M = M\,\mathrm{adj}(M) = \det(M)I. \tag{1}$$

Hence, if $\det(M)$ is invertible, we have $M^{-1} = \frac{1}{\det(M)}\,\mathrm{adj}(M)$.

III. LFSMs

A. Definitions

LFSMs have been studied in [9], [1], [2], [10]. They are a generalization of Linear Feedback Shift Registers, for which the shift structure is removed, i.e. each cell has no privileged neighbor. Let us give a definition of an LFSM (over $\mathbb{F}_2$):

Definition 3.1: A Linear Finite State Machine (LFSM) $\mathcal{L}$, of length n, with k inputs and ℓ outputs consists of:

- A set of n cells, each of them storing a value in $\mathbb{F}_2$. The content of the cells, a binary vector of length n, is denoted $m = (m_0, \ldots, m_{n-1})$ and is called the *state* of the LFSM. We sometimes call the set of these n cells the *register*.
- A *transition function* which is a linear function from $\mathbb{F}_2^n \times \mathbb{F}_2^k$ to $\mathbb{F}_2^n$.
- An *extraction function* which is a linear function from $\mathbb{F}_2^n$ to $\mathbb{F}_2^\ell$.

The behavior of an LFSM is described below:

1 The register is initialized to a state $m^{(0)} \in \mathbb{F}_2^n$ at time $t \leftarrow 0$.
2 The extraction function is used to compute an output vector $v(t) \in \mathbb{F}_2^\ell$ from the state $m^{(t)}$.
3 A new state $m^{(t+1)}$ is computed from the current state $m^{(t)}$ and from a vector $u^{(t)} \in \mathbb{F}_2^k$ input at time t using the transition function. This new state is stored in the register.
4 Execution continues by going back to Step 2, with $t \leftarrow t + 1$.

An LFSM is a class of finite state automaton, for which the set of states is $\mathbb{F}_2^n$ and the transition function is linear. Moreover, an additional function gives the ability to output data. Note also that an LFSM does not terminate as it has no final state.

A given LFSM can be entirely specified by a triplet of $\mathbb{F}_2$-matrices (A, B, C), of respective sizes $n \times n$, $n \times k$ and $\ell \times n$, which describe the transition (A, B) and extraction (C) functions in the following way. Given a state column vector $m^{(t)} \in \mathbb{F}_2^n$ and an input column vector $u^{(t)} \in \mathbb{F}_2^k$, the next state vector $m^{(t+1)}$ and the present output vector $v^{(t)} \in \mathbb{F}_2^\ell$ are expressed by:

$$m^{(t+1)} = Am^{(t)} + Bu^{(t)}, \tag{2}$$

$$v^{(t)} = Cm^{(t)}. \tag{3}$$

For suitable matrices A, B, C, we denote $\mathcal{L}(A, B, C)$ an LFSM with transition and extraction functions given by Equations 2 and 3. For short, we often call A the transition matrix of $\mathcal{L}$ (even when $B \neq 0$) while in fact the transition function depends on both A and B.

The polynomial defined now plays an important role in the theory of LFSMs:

Definition 3.2: Let $\mathcal{L} = (A, B, C)$ be an LFSM. The polynomial $\det(I - XA)$ is called the *connection polynomial* of $\mathcal{L}$. We denoted it $Q_{\mathcal{L}}(X)$ or simply $Q(X)$.

Note that $Q(X) \in \mathbb{F}_2[X]$ has degree at most n (with equality iff $\det(A) \neq 0$). Moreover, $Q(0) = 1$, hence $Q(X)$ has an inverse in the ring $\mathbb{F}_2[[X]]$ of power series. More precisely, $Q(X)^{-1}$ is in $\mathcal{Q}$.

B. Sequences obtained from an LFSM

For each $t \in \mathbb{N}$, an LFSM outputs a vector $v^{(t)} = (v_1^{(t)}, \ldots, v_\ell^{(t)})$ of ℓ bits. For each $i = 1, \ldots, \ell$, we denote $V_i(t_0) = \sum_t^\infty v_i^{(t_0+t)} X^t$ the power series obtained from the sequence $(v_i^{(t)})_{t \geq t_0}$. We also define $V^{(t_0)}$ as the vector $(V_1(t_0), \ldots, V_\ell(t_0))$ of power series. We consider also the series $M_i(t_0) = \sum_t^\infty m_i^{(t_0+t)} X^t$ obtained from the sequence observed in each cell m_i (for $1 \leq i \leq n$), and the vector $M^{(t_0)} = (M_1(t_0), \ldots, M_n(t_0))$ of power series. In a similar way, we define $U^{(t_0)} = (U_1(t_0), \ldots, U_k(t_0))$ from the input sequences.

The sequences $M_i(t_0)$ observed in the register, and the output sequences $V_i(t_0)$ satisfy interesting linear relations (cf. [1], [9], [3]). We provide these relations in the next theorem.

Theorem 3.3: Let $\mathcal{L} = (A, B, C)$ be an LFSM. The vectors $M^{(t_0)}$ and $V^{(t_0)}$ verify:

$$\begin{cases} M^{(t_0)} = \dfrac{\mathrm{adj}(I - XA)}{Q_{\mathcal{L}}(X)}(m^{(t_0)} + XBU^{(t_0)}) \\ V^{(t_0)} = C\dfrac{\mathrm{adj}(I - XA)}{Q_{\mathcal{L}}(X)}(m^{(t_0)} + XBU^{(t_0)}). \end{cases}$$

Proof: For each $t \in \mathbb{N}$, we multiply Equation 2 and Equation 3 by X^t and sum each of them over t. We get

$$M^{(t_0+1)} = AM^{(t_0)} + BU^{(t_0)} \quad (4)$$

$$V^{(t_0)} = CM^{(t_0)}. \quad (5)$$

But $M^{(t_0)} = m^{(t_0)} + XM^{(t_0+1)}$. Hence, with Equation 4 we obtain

$$M^{(t_0)} = X(AM^{(t_0)} + BU^{(t_0)}) + m^{(t_0)}$$

or also $(I - XA)M^{(t_0)} = XBU^{(t_0)} + m^{(t_0)}$. By Equation 1 we obtain the first relation of Theorem 3.3. The second one follows from Equation 5. ∎

Note that, as mentioned before, $1/Q_{\mathcal{L}}(X)$ is a power series. So the expression given for $M^{(t_0)}$ in Theorem 3.3 does not (in general) belong to $\mathbb{F}_2[X]$ but to $\mathbb{F}_2[[X]]$, even if the input U is of finite degree.

Note also that, when the LFSM $\mathcal{L}$ has no input (or more generally when the input U has finite degree), Theorem 3.3 gives expressions for $M_i^{(t_0)}$ and $V_i^{(t_0)}$ as quotients of two polynomials, and so belong to $\mathcal{Q}$, the ring of rational power series.

C. Autonomous LFSMs

An important particular case of LFSMs is the one for which the transition function does not depend on some input, that is to say $B = 0$. Such an LFSM is called an autonomous LFSM. The following Theorem shows that some polynomials p_i (for $1 \leq i \leq n$) related to the components m_i of the state are divided by X modulo $Q(X)$ at each clock cycle.

Theorem 3.4: Let $\mathcal{L}$ be an autonomous LFSM and put $p^{(t)} = \mathrm{adj}(I - XA)m^{(t)}$ (for $t \in \mathbb{N}$). The relation $Xp^{(t+1)} \equiv p^{(t)}$ modulo $Q(X)$ holds, for each t.

Proof: From Equation 2, we have $Xm^{(t+1)} = XAm^{(t)} = -(I - XA)m^{(t)} + m^{(t)}$. Multiplication by $\mathrm{adj}(I - XA)$ gives $Xp^{(t+1)} = -Q(X)m^{(t)} + p^{(t)}$. ∎

D. Similar LFSMs

Two LFSMs defined by two distinct triples (A, B, C) and (A', B', C') may produce the same output. This is the case of *similar* LFSMs, which were defined in [3], [9].

Definition 3.5: Given two LFSMs $\mathcal{L} = (A, B, C)$ and $\mathcal{L}' = (A', B', C')$. $\mathcal{L}$ and $\mathcal{L}'$ are said similar if there exists a non-singular matrix P over $\mathbb{F}_2$ such that:

$$A' = P^{-1}AP, \qquad B' = P^{-1}B, \qquad C' = CP.$$

The matrix P is called the *change basis matrix from $\mathcal{L}$ to $\mathcal{L}'$*.

Theorem 3.6: Let $\mathcal{L}$ and $\mathcal{L}'$ be two similar LFSMs. Assume that their initial state vectors satisfy $m'^{(0)} = P^{-1}m^{(0)}$ and that they have same input ($U^{(0)} = U'^{(0)}$). Then:

1) Both LFSMs $\mathcal{L}$ and $\mathcal{L}'$ have same connection polynomial.
2) $M'^{(0)} = P^{-1}M^{(0)}$. In particular, $m'^{(t)} = P^{-1}m^{(t)}$ holds for each $t \geq 0$.
3) The sequences output by $\mathcal{L}$ and $\mathcal{L}'$ are equal: $V'^{(0)} = V^{(0)}$. In particular, $v'^{(t)} = v^{(t)}$ holds for each $t \geq 0$.

Proof:

1) The first claim results from $\det(I - XA') = \det(I - XP^{-1}AP) = \det(P^{-1}(I - XA)P) = \det(I - XA)$.
2) Let's prove the second claim by recurrence. If $m'^{(t)} = P^{-1}m^{(t)}$ for some t, then Equation 2 gives $P^{-1}m^{(t+1)} = P^{-1}Am^{(t)} + P^{-1}Bu^{(t)} = P^{-1}APm'^{(t)} + P^{-1}Bu^{(t)} = A'm'^{(t)} + B'u'^{(t)} = m'^{(t+1)}$.
3) Finally, using Equation 3, $v'^{(t)} = C'm'^{(t)} = CPP^{-1}m^{(t)} = Cm^{(t)} = v^{(t)}$. This proves the last claim.

∎

E. Classical families of autonomous LFSMs

Different special cases of LFSMs, are well-known and have been extensively studied, with some variations of terminology among different scientific communities. Some results of the theoretic and electronic communities are described in [3], [11], [9] whereas some others coming from the cryptographic community are given in [10], [12], [13], [14]. We gather in this subsection some of these special cases, using notations consistent with the one we used above.

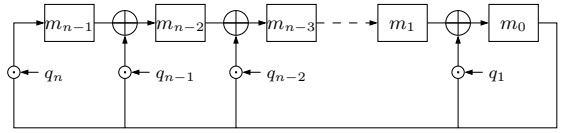
$$T_G = \begin{pmatrix} q_1 & 1 & & & \\ q_2 & & 1 & (0) & \\ \vdots & & (0) & \ddots & \\ q_{n-1} & & & & 1 \\ q_n & 0 & 0 & \cdots & 0 \end{pmatrix}$$

(a) Galois LFSR

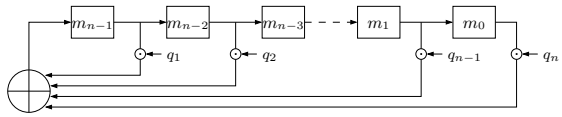
$$T_F = \begin{pmatrix} 0 & 1 & & & \\ 0 & & 1 & (0) & \\ \vdots & & (0) & \ddots & \\ 0 & & & & 1 \\ q_n & q_{n-1} & \cdots & q_2 & q_1 \end{pmatrix}$$

(b) Fibonacci LFSR

Fig. 1. Transition matrices of Galois and Fibonacci LFSRs with connection polynomial $Q(X) = q_n X^n + \cdots + q_1 X + 1$



(a) Galois LFSR



(b) Fibonacci LFSR

Fig. 2. Implementation of Galois and Fibonacci LFSRs with connection polynomial $Q(X) = q_n X^n + \cdots + q_1 X + 1$

The most famous LFSMs special cases are:

- the *Fibonacci Linear Feedback Shift Registers*, also known as *External-XOR LFSR*, or just *LFSR*;
- the *Galois Linear Feedback Shift Registers*, also known as *Internal-XOR LFSR*, or *Canonical LFSR*.

A Galois or Fibonacci LFSR is defined by its connection polynomial because the transition matrix A has a special form and can be deduced from it. The matrices B and C are simple because LFSR have no input and because they output a single bit. The transition matrices for Galois and Fibonacci are shown in Fig. 1. Fig. 2 presents the corresponding implementations.

It can be shown that the matrices T_F and T_G given in Figure 1 are *similar* matrices (because they are "transposed with respect to the second diagonal" one from each other). Hence, the Galois and Fibonacci LFSRs with same connection polynomial are similar LFSMs in the sense of Definition 3.5.

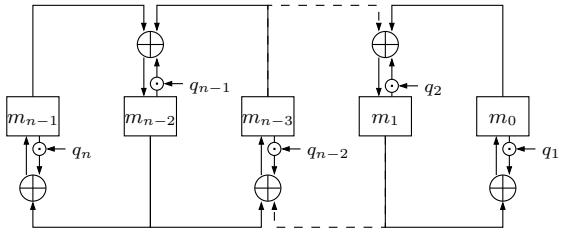
Another special kind of LFSMs is the 3-neighborhood cellular automaton (CA) [3], [11], [15], [16]. These automata are characterized by a tri-diagonal matrix as presented in Fig. 3. They are suitable for hardware implementation.

To cover numerous kind of automata presented in [3], [16], [17], [18], we introduce Ring LFSRs. The cells which store the state are organized in a cyclic shift register. This corresponds to a transition matrix of a particular form:

Definition 3.7: An LFSM $\mathcal{L}$ with transition matrix A is called a *Ring Linear Feedback Shift Register* if the matrix

$$T_{CA} = \begin{pmatrix} q_1 & 1 & & & \\ 1 & q_2 & 1 & (0) & \\ & \ddots & \ddots & \ddots & \\ & (0) & 1 & q_{n-1} & 1 \\ & & & 1 & q_n \end{pmatrix}$$

(a) Transition matrix of a CA



(b) Implementation of a CA

Fig. 3. Transition matrix and implementation of a 3-neighborhood Cellular Automaton

Clock	$\mathcal{L}_0$ *Cells* 7654321 0	$\mathcal{L}_1$ *Cells* 7654321 0	$\mathcal{L}_2$ *Cells* 7654321 0
0	0000000 1	0000000 1	0000000 1
1	1011010 0	1000000 0	1000000 0
2	0101101 0	0100000 0	0100100 0
3	0010110 1	0010000 0	0010010 0
4	1010001 0	1001000 0	1001001 0
5	0101000 1	0100100 0	0100000 1
6	1001110 0	1010010 0	1010000 0
7	0100111 0	0101001 0	0101100 0
8	0010011 1	0010100 1	0010110 0

TABLE I
STATES OF $\mathcal{L}_0$, $\mathcal{L}_1$ AND $\mathcal{L}_2$ DURING 8 CLOCKS.

$A = (a_{i,j})_{0 \leq i,j < n}$ as the following form:

$$\begin{cases} a_{i,i+1} = 1 \text{ for all } 0 \leq i < n-1 \\ a_{n-1,0} = 1 \end{cases}$$

i.e.,

$$A = \begin{pmatrix} & 1 & & (*) & \\ & & \ddots & & \\ & (*) & & \ddots & \\ & & & & 1 \\ 1 & & & & \end{pmatrix}$$

In particular, Galois and Fibonacci LFSRs are special cases of Ring LFSRs.

We detail here a complete example of these automata. Let us consider the primitive connection polynomial $Q(X) = X^8 + X^6 + X^5 + X^3 + 1$ and we denote $\mathcal{L}_0$ the associated Galois LFSR, $\mathcal{L}_1$ the associated Fibonacci LFSR and $\mathcal{L}_2$ a generic Ring LFSR with connection polynomial $Q(X)$. We present their respective transition matrices T_0, T_1 and T_2 in Fig. 4. Fig. 5 shows the implementation of $\mathcal{L}_0$, $\mathcal{L}_1$ and $\mathcal{L}_2$ whereas Table I displays the states of these automata during 8 clocks starting from the same initial state.

The reader can see that from the same initial state `00000001` the output sequences are distinct. However, they

$$T_0 = \begin{pmatrix} 0 & 1 & & & & & & \\ 0 & & 1 & & & (0) & & \\ 1 & & & 1 & & & & \\ 0 & & & & 1 & & & \\ 1 & & & & & 1 & & \\ 1 & & (0) & & & & 1 & \\ 0 & & & & & & & 1 \\ 1 & & & & & & & \end{pmatrix}$$

(a) Galois LFSR

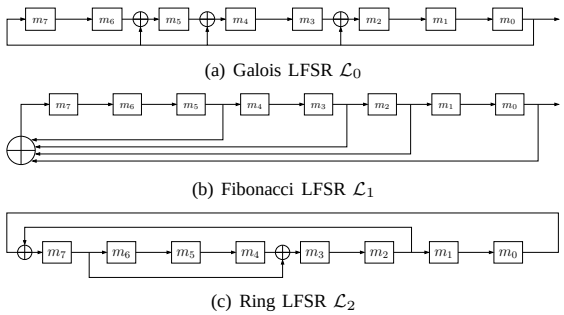
$$T_1 = \begin{pmatrix} & 1 & & & & & & \\ & & 1 & & & (0) & & \\ & & & 1 & & & & \\ & & & & 1 & & & \\ & & & & & 1 & & \\ & & (0) & & & & 1 & \\ & & & & & & & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \end{pmatrix}$$

(b) Fibonacci LFSR

$$T_2 = \begin{pmatrix} & 1 & & & & & & \\ & & 1 & & & (0) & & \\ & & & 1 & & & & \\ & & & & 1 & & & 1 \\ & & & & & 1 & & \\ & & (0) & & & & 1 & \\ & & & & & & & 1 \\ 1 & & 1 & & & & & \end{pmatrix}$$

(c) Ring LFSR

Fig. 4. Transition matrices of $\mathcal{L}_0$, $\mathcal{L}_1$ and $\mathcal{L}_2$



(a) Galois LFSR $\mathcal{L}_0$

(b) Fibonacci LFSR $\mathcal{L}_1$

(c) Ring LFSR $\mathcal{L}_2$

Fig. 5. Three LFSR with connection polynomial $Q(X) = X^8 + X^6 + X^5 + X^3 + 1$

are all a part of the same m-sequence defined by $Q(X) = X^8 + X^6 + X^5 + X^3 + 1$ according to Theorem 3.3. In other words there exists three different polynomials $P_0(X), P_1(X), P_2(X)$ of degrees less than 8 such that the sequences generated by $\mathcal{L}_0$, $\mathcal{L}_1$ and $\mathcal{L}_2$ are respectively $P_0(X)/Q(X)$, $P_1(X)/Q(X)$ and $P_2(X)/Q(X)$.

IV. Rational representation

We introduce a generalization of LFSRs and LFSMs by extending the set of possible coefficients for the transition matrix to rational fractions. This new approach is not only of theoretical interest, but is also an interesting tool for having a more global view of complex circuits and constructing more complex circuits from smaller LFSMs with nice properties. Each coefficient of such a matrix is a rational fraction which represents a small LFSM. The inputs and outputs of each small LFSM are thus used as a part of the full automaton [19].

This new representation allows an easier description of complex circuits with small internal components such as the so-called Windmill generators [4]. These generators are for example used in the stream cipher E0 [5] of Bluetooth.

This rational representation leads to a simpler representation of some circuits with multiple outputs at each iteration or of parallelized versions of LFSRs.

This section is organized as follows: we first focus our analysis on LFSMs with a single input and a single output. Then we introduce the notion of transition matrix with rational coefficients. We demonstrate that the automata built using this new representation essentially produce the same sequences than the classical LFSRs. We give a first example based on this new representation to construct a filtered LFSR automaton. We then focus our work on the case of Windmill generators and give a simpler and more compact definition of such LFSRs. We thus discuss the difficulty of implementing such automata which is not so easy in the general case. Finally, we conclude this section with a concrete example. It consists in a generalization of Windmill generators that allows to construct complex circuits from simpler well designed circuits. These simple circuits are building blocks of a bigger automaton which connects the small components in a circular way. The full circuit inherits good internal properties of the smaller components.

A. LFSMs with a single input and a single output

we are first interested by an LFSM with a single input bit and a single output bit. In this situation, the matrix B is a $n \times 1$ matrix, with a single 1 in position i_0. Likewise, C is a $1 \times n$ matrix, with a single 1 in position j_0.

Set $A' = \mathrm{adj}(I - XA) = (A'_{i,j}(X))$, where the coefficients $A'_{i,j}(X)$ are polynomials, and $Q(X) = \det(I - X.A)$. We can derive from Theorem 3.3, the following relation between the input series $U^{(t)}$ and the output series $V^{(t)}$:

$$V^{(t)} = \frac{X}{Q(X)} CA'BU^{(t)} + \frac{1}{Q(X)} CA'm^{(t)}$$

Note that $CA'B = A'_{j_0,i_0}(X)$ is a polynomial, and $P^{(t)}(X) = CA'm^{(t)}$ is also a polynomial. Setting $R(X) = XA'_{j_0,i_0}(X)$, we can rewrite the previous formula

$$V^{(t)} = \frac{R(X)}{Q(X)} U^{(t)} + \frac{P^{(t)}(X)}{Q(X)}$$

Note that $R(X)$ is independent of the internal state $m^{(t)}$ of the LFSM, and $\frac{P^{(t)}(X)}{Q(X)}$ is uniquely determined by the internal state $m^{(t)}$ of the LFSM.

So up to initial internal values of such LFSM, we can consider that it performs the multiplication of the input by the rational series $R(X)/Q(X)$ (note that, since $Q(X) = \det(I - X.A)$, we have $Q(0) = 1 \neq 0$).

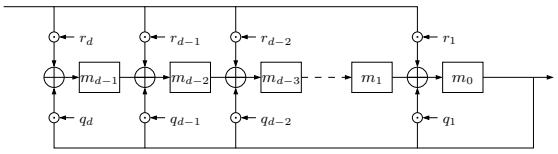


Fig. 6. Implementation of a division/multiplication circuit also known as the serial multiplier/divider for finite fields [20]

Conversely, for a given rational power series $R(X)/Q(X)$, $Q(0) \neq 0$, it is possible to construct many LFSMs which perform the multiplication by $R(X)/Q(X)$.

As an example of such LFSMs, we give in Figure 6 an LFSM with one input and one output which performs the multiplication by $R(X)/Q(X)$ called in the rest of this paper a Galois vane (in reference to a Galois LFSR and a vane of a windmill generator).

The matrix description of this LFSM is:

$$A = \begin{pmatrix} q_1 & 1 & & & \\ q_2 & & 1 & (0) & \\ \vdots & & (0) & \ddots & \\ q_{d-1} & & & & 1 \\ q_d & 0 & 0 & \cdots & 0 \end{pmatrix}, B = \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_{d-1} \\ r_d \end{pmatrix}$$

and $C = (1, 0, \ldots, 0)$.

It is interesting to use some multiplication/division circuits which are not performed by a Galois vane. As an example, we consider the ring LFSR described in Figure 5. The connection polynomial is $Q(X) = X^8 + X^6 + X^5 + X^3 + 1$. Let $T' = \mathrm{adj}(I - XT_2)$, we have $T'_{1,1} = X^6 + X^3 + 1$ and $T'_{4,3} = X^7 + X^5 + X^4 + X^2$. For ${}^tB = C = (1, 0, ..., 0)$, this ring LFSR performs the multiplication by $(X^6 + X^3 + 1)/(X^8 + X^6 + X^5 + X^3 + 1)$ (for more details on this type of circuits, the reader could refer to [21], [22]). For ${}^tB = (0, 0, 0, 1, 0, 0, 0, 0)$ and $C = (0, 0, 1, 0, ..., 0)$, it performs the multiplication by $(X^7 + X^5 + X^4 + X^2)/(X^8 + X^6 + X^5 + X^3 + 1)$. For these two examples, the circuit is simpler than the equivalent one obtained by the Galois vane.

B. Rational Linear Machines

Now, we want to use multiplications by rational power series $R(X)/Q(X)$, with $Q(0) \neq 0$, as internal building blocks in order to construct bigger LFSMs.

Recall that we denote by $\mathcal{Q}$ the ring of rational power series, that is $\{P(X)/Q(X) \in \mathbb{F}_2[[X]] \mid P(X), Q(X) \in \mathbb{F}_2[X], Q(0) \neq 0\}$.

Definition 4.1: A Rational Linear Machine (RLM) $\mathcal{L}$ with k-bit input, ℓ-bit output and length n over $\mathcal{Q}$ is a triplet of matrices (A, B, C) over $\mathcal{Q}$, of respective sizes $n \times n$, $n \times k$, $\ell \times n$. Given the current state vector $(m^{(t)}, c^{(t)}) \in \mathcal{M}_{n,1}(\mathbb{F}_2) \times \mathcal{M}_{n,1}(\mathcal{Q})$ and input vector $u^{(t)} \in \mathcal{M}_{k,1}(\mathbb{F}_2)$. The next state vector $(m^{(t+1)}, c^{(t+1)})$ and the present output vector $v^{(t)} \in \mathcal{M}_{\ell,1}(\mathbb{F}_2)$ are expressed as:

$$\begin{cases} m^{(t+1)} &= Am^{(t)} + c^{(t)} + Bu^{(t)} \bmod X \\ c^{(t+1)} &= Am^{(t)} + c^{(t)} + Bu^{(t)} \operatorname{div} X \\ v^{(t)} &= Cm^{(t)} \end{cases}$$

where $P(X) \operatorname{div} X = \frac{P(X) - (P(X) \bmod X)}{X}$.

As previously we are able to describe the output sequences:

Theorem 4.2: Let $\mathcal{L} = (A, B, C)$ a RLM. The vector $M^{(t)}$ satisfy the relation:

$$M^{(t)} = (I - XA)^{-1} \left(m^{(t)} + Xc^{(t)} + XBU^{(t)} \right)$$

Proof: With the previous notations we have the following relations:

$$M^{(t+1)} = AM^{(t)} + c^{(t)} + BU^{(t)} \quad (6)$$

$$M^{(t)} = XM^{(t+1)} + m^{(t)} \quad (7)$$

Equation 6 is by Definition 4.1. Equation 7 comes from the Definition of $M^{(t)}$. It leads to the following relation:

$$(I - XA)M^{(t_0)} = m^{(t_0)} + Xc^{(t_0)} + XBU^{(t)}$$

Note that $(I - XA)$ is invertible in $\mathcal{M}_n(\mathcal{Q})$. This leads to $M^{(t_0)} = (I - XA)^{-1}(m^{(t_0)} + Xc^{(t_0)} + XBU^{(t)})$ in $\mathcal{Q}$. ■

C. Rational Linear Finite State Machines

In order to focus the attention on some applications, and for a better understanding of the significance of Theorem 4.2, we focus in this Section on the study of RLM with no input. Moreover, we try to limit the domain of the "carries" register c in order to ensure that the machine is a finite state machine. We suppose in the sequel that $B = 0$, i.e. there is no input.

In order to restrict RLM to finite state machines, we have to look at the evolution of "internal memories" $c^{(t)}$ in more details. Let $A_{i,j} = P_{i,j}(X)/Q_{i,j}(X)$ be the expression of a coefficient of the matrix A as a quotient of two polynomials. For a fixed row i we can compute the polynomial $Q_i(X) = \mathrm{lcm}(Q_{i,1}(X), \ldots, Q_{i,n}(X))$. So we can normalize the rational representations as follows: $A_{i,j} = R_{i,j}(X)/Q_i(X)$. For each row i we define the following finite subset of $\mathcal{Q}$: $W_i = \{R(X)/Q_i(X) \mid \deg(R(X)) < \max_j(\deg(R_{i,j}(X)))\}$. Finally we define $W = \prod_{i=1}^{n} W_i \subset \mathcal{Q}^n$. Note that W is a finite set. The following proposition shows that it is a "reasonable" set for the values of the internal memories;

Proposition 4.3: Suppose that at time t_0, $c^{(t_0)}$ is in W, then for any $t \geq t_0$, $c^{(t)}$ is in W.

Proof: Let $\mu^{(t+1)} = Am^{(t)} + c^{(t)}$. From the definition of a RLM, we have $m^{(t+1)} = \mu^{(t+1)} \bmod X$ and $c^{(t+1)} = \mu^{(t+1)} \operatorname{div} X$.

If we consider the i-th row of A, we obtain $\mu_i^{(t+1)} = \sum_{j=1}^{n} m_j^{(t)} R_{i,j}(X)/Q_i(X) + c_i^{(t)}$. So under the condition $c_i^{(t)} \in W_i$, $\mu_i^{(t+1)}$ can be expressed as a rational fraction of the form R'_i/Q_i and $\deg(R'_i) \leq \max_j(\deg(R_{i,j}(X)))$, this implies $c^{(t+1)} \in W_i$. ■

Following this result we want to limit the "carries" part of a RLM to the domain W. So we give the following definition for RLFSMs, which is a true finite state machine.

Definition 4.4: A Rational Linear Finite State Machine (RLFSM) with ℓ-bit output and length n over $\mathcal{Q}$ is a finite state automaton defined by a pair (A, C) of matrices over $\mathcal{Q}$, with respective sizes $n \times n$ and $\ell \times n$. The space of states of this automaton is $\mathbb{F}_2^n \times W$ where W is defined from A as previously explained, the transition and extraction functions at time t are

defined by: if the automaton is in the state $(m^{(t)}, c^{(t)})$ at time t and $v^{(t)}$ is the output at time t, then

$$\left\{ \begin{array}{rcl} m^{(t+1)} & = & Am^{(t)} + c^{(t)} \bmod X \\ c^{(t+1)} & = & Am^{(t)} + c^{(t)} \operatorname{div} X \\ v^{(t)} & = & Cm^{(t)} \end{array} \right.$$

Now, we want to characterize in more details the output of a RLFSM. Set $G(X) = \prod_{i=1}^{n} Q_i(X)$. We have $A = \frac{1}{G(X)} A'$, where A' is a matrix with polynomial coefficients.

From the definition of A', we have $I - XA = G(X)^{-1}(G(X)I - XA')$. We deduce $(I - XA)^{-1} = G(X)(G(X)I - XA')^{-1}$.

Set $T(X) = \det(G(X)I - XA')$. Note that $T(X)$ is a polynomial. We have $(G(X)I - XA')^{-1} = \frac{1}{T(X)} \operatorname{adj}(G(X)I - XA')$, so $(I - XA)^{-1} = \frac{G(X)}{T(X)} \operatorname{adj}(G(X)I - XA')$. The important point is that $\operatorname{adj}(G(X)I - XA')$ is a matrix with polynomial coefficients.

We can easily deduce the rational form of the output of a RLFSM

Proposition 4.5: Let $\mathcal{L}$ be a RLFSM defined by a transition matrix A and any output matrix C. Set $T(X) = \det(G(X)(I - XA))$. The output sequences $V_i^{(t)}$ are rational power series of the form $P_i(X)/T(X)$.

Proof: This result comes from the formula

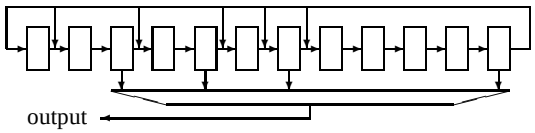
$$\begin{array}{rcl} M^{(t)} & = & (I - XA)^{-1}(m^{(t)} + Xc^{(t)}) \\ & = & \frac{G(X)}{T(X)} \operatorname{adj}(G(X)I - XA')(m^{(t)} + Xc^{(t)}). \end{array}$$

Indeed, the denominators of the coefficients of the matrix $(I - XA)^{-1}$ are some divisors of $T(X)$, $m^{(t)}$ is a binary vector and $c^{(t)} \in W$ is such that $G(X)^n X c^{(t)}$ is a polynomial vector. ∎

Note that the rational power series $P_i(X)/T(X)$ are *a priori* not irreducible. In practice, the numerator is often the polynomial $Q(X)$ such that $Q(X)/P(X)$ is the irreducible rational representation of $\det(I - XA)$.

D. A first example

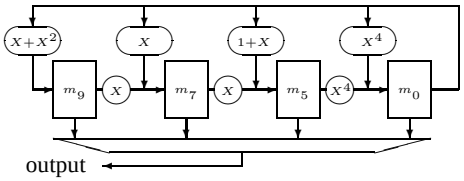
We consider a filtered LFSR in Galois mode of size $n = 12$ with connection polynomial $Q(X) = 1 + X^5 + X^6 + X^7 + X^9 + X^{11} + X^{12}$, filtered by a Boolean function in cells m_0, m_5, m_7 and m_9.



If we are interested only on the filtered output bits, this LFSR can be described by a RLFSM with the matrix

$$A' = \begin{pmatrix} X^4 & X^4 & 0 & 0 \\ 1+X & 0 & X & 0 \\ X & 0 & 0 & X \\ X+X^2 & 0 & 0 & 0 \end{pmatrix}$$

This matrix leads to a new representation of this RLFSM:



Let $B = (I - XA')^{-1}$, and $Q(X) = \det(I - XA') = X^{12} + X^{11} + X^9 + X^7 + X^6 + X^5 + 1$. Then the value of B is

$$B = \frac{1}{Q(X)} \begin{pmatrix} P_{1,1} & P_{1,2} & P_{1,3} & P_{1,4} \\ P_{2,1} & P_{2,2} & P_{2,3} & P_{2,4} \\ P_{3,1} & P_{3,2} & P_{3,3} & P_{3,4} \\ P_{4,1} & P_{4,2} & P_{4,3} & P_{4,4} \end{pmatrix}$$

with $P_{1,1}(X) = 1$, $P_{1,2}(X) = X^5$, $P_{1,3}(X) = X^7$, $P_{1,4}(X) = X^9$, $P_{2,1}(X) = X^7 + X^6 + X^4 + X^2 + X$, $P_{2,2}(X) = X^5 + 1$, $P_{2,3}(X) = X^7 + X^6 + X^5 + 1$, $P_{2,4}(X) = X^9 + X^8 + X^7 + X^2$, $P_{3,1}(X) = X^5 + X^4 + X^2$, $P_{3,2}(X) = X^{10} + X^9 + X^7$, $P_{3,3}(X) = X^7 + X^6 + X^5 + 1$, $P_{3,4}(X) = X^9 + X^8 + X^7 + X^2$, $P_{4,1}(X) = X^3 + X^2$, $P_{4,2}(X) = X^8 + X^7$, $P_{4,3}(X) = X^{10} + X^9$ and $P_{4,4}(X) = X^9 + X^7 + X^6 + X^5 + 1$.

If we denote by $(a_0, \ldots, a_{11})$ the initial state at time $t = 0$ of the binary LFSR, then, the initial state of our RLFSM is $m^{(0)} = (a_0, a_5, a_7, a_9)$ and $c^{(0)} = (a_1 + a_2X + a_3X^2 + a_4X^3, a_6, a_8, a_{10} + a_{11}X)$ and the sequences in output are

$$\frac{a_0P_{1,1}(X) + a_5P_{1,2}(X) + a_7P_{1,3}(X) + a_9P_{1,4}(X)}{Q(X)} + \frac{(a_1 + a_2X + a_3X^2 + a_4X^3)X}{Q(X)},$$
$$\frac{a_0P_{2,1}(X) + a_5P_{2,2}(X) + a_7P_{2,3}(X) + a_9P_{2,4}(X) + a_6X}{Q(X)},$$
$$\frac{a_0P_{3,1}(X) + a_5P_{3,2}(X) + a_7P_{3,3}(X) + a_9P_{3,4}(X) + a_8X}{Q(X)},$$
$$\frac{a_0P_{4,1}(X) + a_5P_{4,2}(X) + a_7P_{4,3}(X) + a_9P_{4,4}(X) + (a_{10} + a_{11})X}{Q(X)}.$$

E. Application to windmill generators

Windmill generators can be defined as LFSMs with no input and several outputs. They have been introduced in [4] as a cyclic cascade connection of $v \geq 1$ LFSMs. Each of these LFSMs is called a vane of the windmill. The classical representation of those LFSMs is the Fibonacci one. However, in the rest of this section, we show them using the equivalent Galois representation because it is more suitable for a better understanding. Windmill generators are characterized by their feedback and feedforward connections. These feedback and feedforward connections are identical for all vanes, but the lengths of the LFSMs may be different as they can be shifted in different LFSMs. Fig. 6 presents a generic vane in Galois mode.

Windmill generators were introduced to achieve parallel generation of sequences [19]. Consider a sequence $S = (s_n)_{n \in \mathbb{N}}$. While a classical automaton outputs s_0 at the first clock, s_1 at the second, and so on, a parallel automaton outputs v bits at each clock: $(s_0, s_1, \ldots, s_{v-1})$ at the first clock, $(s_v, \ldots, s_{2v-1})$ at the second, etc. More precisely a parallel automaton has v outputs and products the sequences $S^i := (s_{nv+i})_{n \in \mathbb{N}}$ where $0 \leq i < v$. Note that our study focus

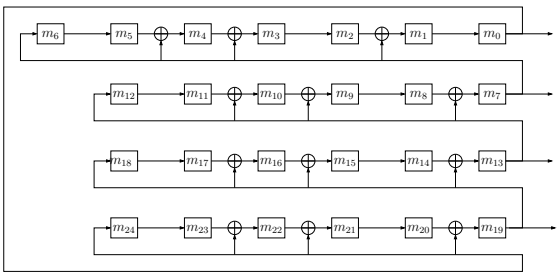


Fig. 7. A windmill with only feedforward connections.

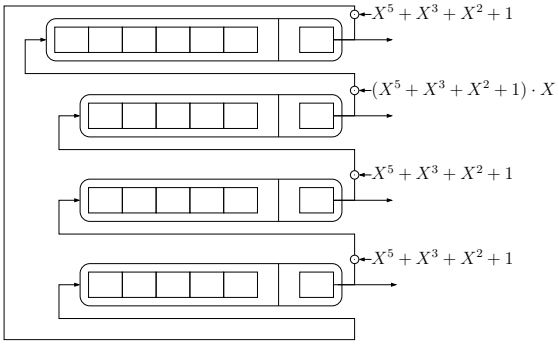


Fig. 8. A windmill in rational representation.

on characterizing the sequences S^i and not the reconstructed sequence S.

Consider the windmill presented in Fig. 7 which is the one used in the stream cipher E0 [5] (up to the Galois/Fibonacci representation). It is constituted of one vane of length 7 and three identical vanes of length 6. No feedback connection appears. Feedforward connections appear, for example from cell m_{13} to cells m_{12}, m_{10}, m_9 and m_7.

Until now, only windmill generators with a single vane repeated several times have been studied. We generalize this definition allowing different vanes in a windmill. We also give a new description of this windmill which is more compact. More precisely, using the example, we want to consider output sequences of cells m_0, m_7, m_{13} and m_{19}, and characterize each vane by a polynomial. This leads to the interpretation presented in Fig. 8.

With this definition the LFSM described in Figure 8 as the following transition matrix:

$$(X^5+X^3+X^2+1)\cdot\begin{pmatrix}0&X&0&0\\0&0&1&0\\0&0&0&1\\1&0&0&0\end{pmatrix}$$

We give in Table II the values of $m^{(t)}$ and $c^{(t)}$ during 8 clocks.

According to Definition 4.4, windmills as introduced by Smeets and Chambers [4] agree with the following definition:

Definition 4.6: A windmill generator with polynomials $\alpha(X), \beta(X)$ with $\beta(0)\neq 0$ and v vanes is an LFSR of length v with matrix A over $\mathbb{F}_2[[X]]$ of the form:

$$\begin{pmatrix}0&\frac{\alpha(X)}{\beta(X)}X^{i_0}&&(0)\\\vdots&\ddots&\ddots&\\0&(0)&\ddots&\frac{\alpha(X)}{\beta(X)}X^{i_{v-2}}\\\frac{\alpha(X)}{\beta(X)}X^{i_{v-1}}&0&\dots&0\end{pmatrix}$$

where $0\leq i_0,\dots,i_{v-1}$.

With this representation each row represents a vane of the windmill. In particular, as described in the following section the length of the vane j is equal to $\max(\deg(\alpha(X)X^{i_j}),\deg(\beta(X)))$.

By a straightforward calculus, we obtain $\det(I-XA)=X^n\left(\alpha(X)/\beta(X)\right)^v+1$, where $n=i_0+\cdots+i_{v-1}$. Set $Q(X)=X^n\alpha(X)^v+\beta(X)^v$, it becomes $\det(I-XA)=Q(X)/\beta(X)^v$. The sequences $M_i^{(t)}$ observed in the output of this RLFSM are of the form $P_i(X)/Q(X)$. The main result on windmill generators (c.f. [4]) is the fact that there exists a permutation σ of $\{0,\dots,v-1\}$ such that the series $S(X)=\sum_t(\sum_{i=0}^{v-1}m_i(t)X^{\sigma(i)})X^{vt}$ is a rational power series of the form $P(X)/Q(X^v)$. In other words, a windmill generator is able to output in parallel at each iteration v consecutive values of a rational power series. The most interesting case is the one where $Q(X^v)$ is a primitive polynomial.

Our polynomial approach gives a more synthetic point of view on these windmill generators. In particular, it shows that the windmill properties (i.e. the parallel generation of a given m-sequence) is independent of the implementation of the vanes. This implementation can be made with Fibonacci vanes as in the original version, or with Galois vanes as presented previously or with ring vanes with better diffusion delay as we will see in the next section.

F. Implementation of RLFSMs

In our previous examples, the starting point was a binary circuit, or a RLFSM with a particular structure for its matrix. The converse problem is "how to construct an efficient implementation from a given transition matrix A of a RLFSM". We show on two examples that this task is not so easy.

1) A first example: Consider the RLFSM $\mathcal{L}^1$ defined by the following transition matrix:

$$A=\begin{pmatrix}\frac{X^2}{X^3+1}&\frac{X}{X^2+X+1}\\1&0\end{pmatrix}$$

We compute $(I-XA)^{-1}$ to characterize the output sequences:

$$(I-XA)^{-1}=\begin{pmatrix}\frac{X^3+1}{X^4+X^3+1}&\frac{X^3+X^2}{X^4+X^3+1}\\\frac{X^4+X}{X^4+X^3+1}&\frac{1}{X^4+X^3+1}\end{pmatrix}$$

Figure 9 presents an implementation of this automaton built upon three LFSMs. One for each nonzero coefficient in A. These LFSMs are built using a Galois vane architecture as presented in Fig. 6.

$Clock$	$m_0^{(t)}$	$m_1^{(t)}$	$m_2^{(t)}$	$m_3^{(t)}$	$c_0^{(t)}$	$c_1^{(t)}$	$c_2'^{(t)}$	$c_3^{(t)}$
0	1	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	X^4+X^2+X
2	0	0	1	0	0	0	X^4+X^2+X	X^3+X+1
3	0	1	0	1	0	X^4+X^2+X	X^3+X+1	X^2+1
4	0	0	0	1	$X^5+X^3+X^2+1$	X^3+X+1	X^4+X+1	X
5	1	1	0	0	X^4+X^2+X	X^2+1	$X^4+X^3+X^2+X+1$	1
6	0	1	1	0	X^5+X^2+X	X	X^3+X^2+X+1	X^4+X^2+X
7	0	1	1	0	$X^5+X^4+X^3+X^2+X$	X^4+X^2+X+1	X^2+X+1	X^3+X+1
8	0	0	1	1	X^5+X^4+X	$X^4+X^3+X^2+1$	$X+1$	X^2+1

TABLE II
STATES OF FIGURE 8 DURING 8 CLOCKS.

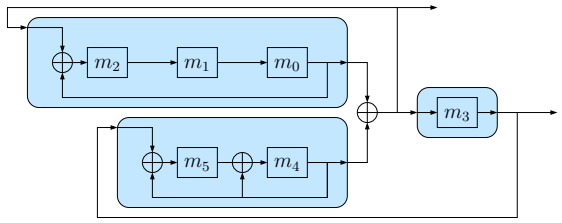


Fig. 9. First implementation of $\mathcal{L}^1$.

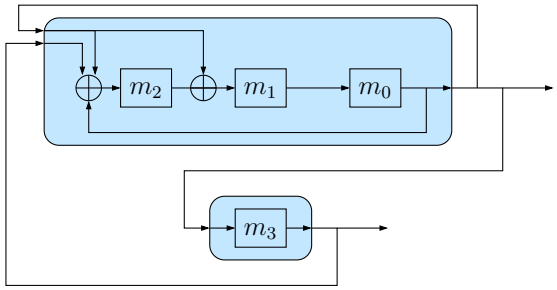


Fig. 10. Second implementation of $\mathcal{L}^1$.

Note that, according to the notation of Figure 9, $\mathcal{L}^1$ can be expressed as the LFSM $(A', 0, C')$ with:

$$A' = \begin{pmatrix} 0&1&0&0&0&0\\ 0&0&1&0&0&0\\ 0&0&0&0&1&0\\ 1&0&0&0&1&0\\ 0&0&0&0&1&1\\ 0&0&0&1&1&0 \end{pmatrix},\ C' = \begin{pmatrix} 1&0&0&0&1&0\\ 0&0&0&1&0&0 \end{pmatrix}$$

In particular, we have the following relations according to Theorem 3.3:

$$V^{(t)} = \frac{1}{X^4+X^3+1} \times \begin{pmatrix} 1 & X & X^2 & X^3+X^2 & X+1 & X^2+X \\ X & X^2 & X^3 & 1 & X^2+X & X^3+X^2 \end{pmatrix} m^{(t)}$$

This implementation is not optimal because it requires seven memories cells while four are enough (it outputs sequences of the form $P(X)/(X^4+X^3+1)$ with $\deg P(X) < 3$). In particular, $\det(I - XA') = X^6+X^3+X^2+X+1$, i.e., this automaton could output m-sequences of the form $P(X)/(X^6+X^3+X^2+X+1)$ using a different matrix C' because $X^6+X^3+X^2+X+1$ is primitive.

A better implementation is given considering one LFSM per line. To do so, note that $\frac{X}{X^2+X+1} = \frac{X^2+X}{X^3+1}$. This leads to the implementation presented in Figure 10.

As previously this leads to the relation:

$$V^{(t)} = \begin{pmatrix} \frac{1}{X^4+X^3+1} & \frac{X}{X^4+X^3+1} & \frac{X^2}{X^4+X^3+1} & \frac{X^3+X^2}{X^4+X^3+1} \\ \frac{X}{X^4+X^3+1} & \frac{X^2}{X^4+X^3+1} & \frac{X^3}{X^4+X^3+1} & \frac{1}{X^4+X^3+1} \end{pmatrix} m^{(t)}.$$

2) Second example: Consider the RLFSM $\mathcal{L}^2$ defined by the following transition matrix:

$$A = \begin{pmatrix} \frac{X+1}{X^3+X+1} & \frac{X}{X^2+X+1} & 0 \\ X^3+X^2 & X^2 & 1 \\ 0 & \frac{X+1}{X^2+X+1} & 0 \end{pmatrix}$$

Figure 11 presents an implementation of this automaton built upon six LFSMs. One for each nonzero coefficient in A. These LFSMs are built using a Galois vane architecture as presented in Figure 6.

Note that, according to the notation of Figure 11, $\mathcal{L}^2$ can be expressed as the LFSM $(A', 0, C')$ with:

$$A' = \begin{pmatrix} 0&1&0&1&0&0&0&0&0&0&0&0&0&0&0\\ 1&0&1&1&0&0&0&0&0&0&0&0&0&0&0\\ 1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&1&1&1&0&0&0&1&0&0&1&0&0\\ 0&0&0&1&0&1&0&0&0&1&0&0&1&0&0\\ 0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&1&0&0&0&0&0&0&0\\ 1&0&0&1&0&0&0&0&1&0&0&0&0&0&0\\ 1&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&1&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\ 0&0&0&0&0&1&0&0&0&1&0&0&1&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0&0&1&0\\ 0&0&0&0&0&1&0&0&0&0&1&0&0&1&1\\ 0&0&0&0&0&1&0&0&0&1&0&0&1&1&0 \end{pmatrix}$$

and

$$C' = \begin{pmatrix} 1&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&1&0&0&0&0&1&0&0&1&0\\ 0&0&0&0&0&0&0&0&0&0&0&0&0&1&0 \end{pmatrix}$$

This implementation is not optimal because it requires fifteen memories cells while nine are enough because $\deg(\det(I - XA)) = 9$. In particular, $\deg(\det(I - XA')) = 11$.

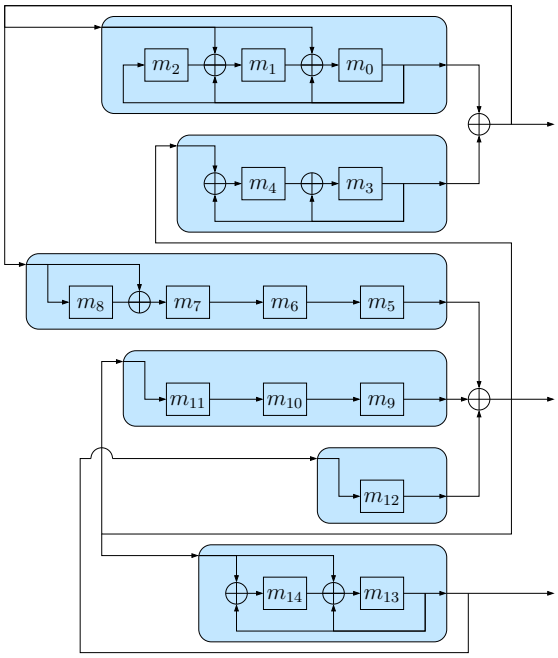


Fig. 11. First implementation of $\mathcal{L}^2$.

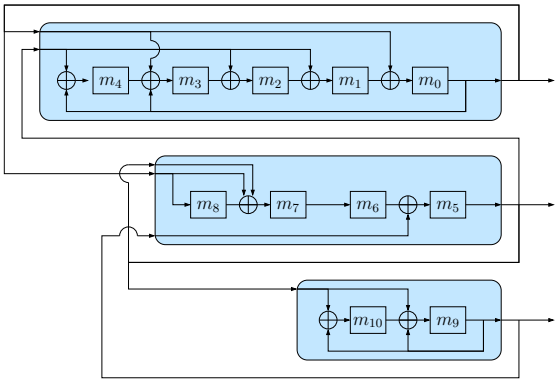


Fig. 12. Second implementation of $\mathcal{L}^2$.

A better implementation is given considering one LFSM per line. This leads to the implementation presented in Figure 12.

This implementation is still not optimum because it requires eleven memory cells. This comes from the fact that in the matrix A, two terms with identical denominator appears in the same column: $\frac{X}{X^2+X+1}$ and $\frac{X+1}{X^2+X+1}$. More precisely, $\det(I-XA') = (X+1)(X^2+X+1)(X^8+X^7+X^5+X^4+X^3+X^2+1)$. Thus, the automaton could be implemented using the nine cells equivalent with the polynomial $(X+1)(X^8+X^7+X^5+X^4+X^3+X^2+1)$ which is reducible and thus not primitive whereas the last factor disappears inside the automaton itself.

G. A practical example of application

The rational representation is a theoretical tool that provides a global view on the LFSRs design, as seen for the case of windmill generators. However, previous examples have shown that starting from a circuit under rational representation to obtain an optimal implementation is not a simple task.

In the example given here, we generalize the windmills generators through particular series circuits. We limit our study with an example built on 3 circuits but the generalization of this method is straightforward.

Let $A_1(X) = P_1(X)/Q_1(X)$, $A_2(X) = P_2(X)/Q_2(X)$ and $A_3(X) = P_3(X)/Q_3(X)$ be 3 elements of $\mathcal{Q}$. We consider the rational LFSR with transition matrix

$$T = \begin{pmatrix} 0 & A_1 & 0 \\ 0 & 0 & A_2 \\ A_3 & 0 & 0 \end{pmatrix}.$$

We have $\det(I-XT) = 1 - X^3A_1A_2A_3 =$ i.e. $\det(I-XT) = Q(X)/(Q_1(X)Q_2(X)Q_3(X))$ with $Q(X) = Q_1(X)Q_2(X)Q_3(X) + X^3P_1(X)P_2(X)P_3(X)$. The associated automaton computes rational series of the form $(P(X)Q_1(X)Q_2(X)Q_3(X))/Q(X)$ for some polynomial $P(X)$ depending on the initial state of the automaton.

Following the examples introduced in Figure 5 and in Section IV-A, we choose $A_1(X) = A_2(X) = (X^6+X^3+1)/(X^8+X^6+X^5+X^3+1)$ and $A_3(X) = (X^7+X^5+X^4+X^2)/(X^8+X^6+X^5+X^3+1)$. The connection polynomial (i.e. the numerator of $\det(I-XT)$) is $Q(X) = X^{24}+X^{21}+X^{16}+X^9+X^7+X^3+1$. This polynomial is primitive, so the automaton produces m-sequences.

For a practical implementation, we can replace the Galois vanes associated to $A_1(X)$, $A_2(X)$ and $A_3(X)$ by the ring vanes presented in Section IV-A.

This leads to a classical binary LFSR with transition matrix

$$T_r = \begin{pmatrix} T_2 & 0 & E_{1,4} \\ E_{1,1} & T_2 & 0 \\ 0 & E_{3,1} & T_2 \end{pmatrix}.$$

Where T_2 is the 8×8 matrix of the ring LFSR given in Figure 4 and where $E_{i,j}$ is the 8×8 matrix with only one 1 in position (i,j). The matrices $E_{i,j}$ represent the connections between the 3 circuits. For example, the matrix $E_{1,4}$ corresponds to the input 1 of the first ring LFSR and the output 4 of the third LFSR.

Note that $\det(I-XT_r) = Q(X) = X^{24}+X^{21}+X^{16}+X^9+X^7+X^3+1$.

Suppose now that we prefer an implementation with Galois vanes as internal blocks. The matrix of the Galois vane is the matrix T_0 given in Figure 4. The multiplication by $A_1(X)$ is performed using ${}^tB_1 = (1,0,0,1,0,0,1,0)$ in input and $C_1 = (1,0,0,0,0,0,0,0)$ for output. In the same way, we obtain $B_2 = B_1$, ${}^tB_3 = (0,0,1,0,1,1,0,1)$ and $C_2 = C_3 = C_1$. So the equivalent binary circuit is then

$$T_g = \begin{pmatrix} T_0 & 0 & B_1C_3 \\ B_2C_1 & T_0 & 0 \\ 0 & B_3C_2 & T_0 \end{pmatrix}.$$

As we will see in the next section the automaton corresponding to the matrix T_r has many nice properties compared to the classical ones obtained from the Galois LFSR. In particular, it needs 9 connections compared to 19 for the second one.

This example shows that the rational representation allows to separate the global design of the automaton from the choices of the hardware (or software) implementation.

The method presented in this example can be directly generalized to all Windmill generators and potentially leads to better practical implementations.

V. Design of efficient LFSRs for both hardware and software cryptographic applications

In this section, we specialize our work on autonomous LFSMs, in particular on LFSRs and their dedicated use for cryptographic applications.

A general purpose of cryptography is to design primitives that are both efficient in hardware and software because such primitives must run on all possible supports, from RFID tags to super-calculators. Thus, cryptographers must keep in mind, when they design cryptosystems, the very wide range of targets on which cryptosystems must be rapid and efficient. As proof, the Rijndael algorithm chosen as the AES [23] in 2001 was one of the more efficient algorithm in hardware and in software among the finalists of the AES competition.

Thus, designing well-chosen dedicated LFSMs efficient both in hardware and in software has direct consequences on the celerity of the cryptosystems which use such primitives as building blocks. Among cryptographic primitives that use LFSMs, we could cite the most famous case: the stream ciphers. Many stream ciphers - such as E0 [5], SNOW [7] or the finalists SOSEMANUK [24] and Grain v1 [25] of the eStream project [26] - filter the content of one or many LFSMs to output pseudo-random bits. LFSMs could also be used as diffusion layer of a block cipher as proposed in [27]. More recently, a particular LFSM combined with two NLFSRs (Non-Linear Feedback Shift Registers) has been proposed in [28] at CHES 2010 as the building block of a lightweight hash function named Quark. Well designing LFSMs with good criteria is therefore crucial for symmetric key cryptography.

In this section, we first introduce the required design criteria that must be fulfilled by an LFSM when used in cryptographic applications. We then extend the traditional concept of diffusion (well-known in the block cipher context) to the case of LFSMs. This leads to define a new criterion for good LFSMs choices for cryptographic applications which is defined as the counterpart of the Shannon diffusion concept [29].

Then, we present previous works on LFSMs for hardware and software cryptographic applications. These automata have been widely studied [1], [2], [4], [10], [30], [6] and practical constructions have emerged. We finally propose an efficient construction dedicated to hardware and a second one dedicated to software. This software construction is also efficient in hardware.

A. Design criteria

We focus our design analysis on two important properties. The first one characterizes the kind of sequences that are required for cryptographic applications whereas the second one tries to formalize the notion of diffusion delay in the context of LFSRs.

1) m-sequences: As introduced in Section II, m-sequences are particular linear recurring sequences with good properties [1], [10]. For example, we give some properties for m-sequences of degree n over $\mathbb{F}_2$:

- an m-sequence is balanced: the number of 1 is one greater than the number of 0 (considering one period).
- an m-sequence has the run property: a run is a subsequence of 1 or 0 followed and followed by 0 or 1. Half of the runs are of length 1, a quarter of length 2, an eighth of length 3, etc. up to the 1-run of length n.
- an m-sequence is a punctured De Bruijn sequence.
- an m-sequence has the (ideal) two-level autocorrelation function where the autocorrelation function for a binary sequence a is defined as $C_a(\tau) = \sum_{i=0}^{N-1}(-1)^{a_{i+\tau}+a_i}$ where N is the period of the sequence. This function satisfies for an m-sequence: $C_\tau = N$ if $\tau = 0 \mod N$ and $C_\tau = K$ if $\tau \neq 0 \mod N$ (where K is a constant equal to -1 if N is odd and to 0 is N even).
- an m-sequence has maximum period: an m-sequence verifying a linear relation of degree n has a period of $2^n - 1$.

In the sequel, we are specially interested in LFSMs having a primitive connection polynomial and producing m-sequence which are the ones classically used in cryptography. In particular, all our examples satisfy this condition. However, most of the results remains true without this hypothesis.

2) Diffusion delay: The concept of diffusion for a cipher was introduced by C. Shannon in [29] as the dissipating effect of the redundancy of the statistical structure of a message M. This concept is directly linked with the Avalanche effect defined by H. Feistel in [31] which is a desirable property of cryptographic algorithms, typically block ciphers and cryptographic hash functions. The Avalanche effect means that if an input is changed slightly, the corresponding output must change significantly. In the case of block ciphers, such a small change in either the key or the plaintext should cause a drastic change in the ciphertext.

Two precise notions could be directly derived: the strict avalanche criterion (SAC) and the bit independence criterion (BIC). The strict avalanche criterion (SAC) is a generalization of the avalanche effect. It is satisfied if, whenever a single input bit is complemented, each of the output bits changes with a 50% probability [32]. The bit independence criterion (BIC) states that output bits j and k should change independently when any single input bit i is inverted, for all i, j and k.

When focusing on m-sequences, the measure of diffusion capacity is usually studied through the notions of correlation, auto-correlation and cross-correlation (see [33] for more details). The correlation of two binary m-sequences $\alpha = (a_1, \cdots a_n)$ and $\beta = (b_1, \cdots b_n)$ is measured as $C(\alpha, \beta) = \frac{1}{n}(A - D)$ where A is the number of times for i from 1 to n, that a_i and b_i agree and D is the number of times that a_i and b_i disagree. The auto-correlation of a given binary sequence has already been defined in the previous subsection. It represents the similarity between a sequence and its phase shift. The

cross-correlation is defined as $C_{\alpha,\beta}(\tau) = \sum_{i=0}^{N}(-1)^{a_{i+\tau}+b_i}$ when $q = 2$ for two periodic binary sequences α of period s and β of period t with $N = \text{lcm}(s,t)$ (for the case $q > 2$ the reader could refer to [33]).

Thus, in this part, we introduce a slightly different definition of diffusion of an LFSM to more precisely capture the behavior of the beginning of a sequence. This parameter measures the time needed to mix the content of the cells of an automaton. It could be expressed as the minimal number of clocks needed such that any memory cell has been influenced by any other.

Definition 5.1: Let $\mathcal{L} = (A, 0, C)$ be an LFSM. Denote by G the graph defined by the adjacency matrix A^t, i.e., if $a_{i,j} \neq 0$ then there exists a directed edge from vertex j and to vertex i. The diffusion delay is equal to the diameter of G.

This parameter does not focus on the output sequence of an LFSM but on the sequences produced INSIDE the register itself (i.e. we look at the sequences $(m_0(t), \cdots, m_{n-1}(t), \cdots)$) and thus is relied on the implementation of the automaton.

In a general point of view, if we take a random graph with n vertices, the average value of its diffusion delay is $\sqrt{n}$ as shown in [34]. For a complete graph, the diffusion delay parameter is optimal and is equal to 1, however complete graphs do not produce good sequences as the corresponding determinant $\det(I-AX)$ (where A is the matrix representation of the complete graph) is equal to $X + 1$ if n is odd and 1 otherwise and thus could not produce sufficiently large m-sequences. Moreover, for a complete graph, from the circuit point of view, as the matrix of such graph as n^2 non-zero terms, this means that the representation circuit has $n^2 - n$ xors. In the same way, the required number of xors for a circuit representing a random graph is about $n^2/2$. But, for cryptographic applications with efficient implementations, we look at circuits with good properties and with about $n/2$ xors which correspond with matrices with a binary weight equal to $3n/2$. Thus, we are far from circuits of complete or random graphs.

So, we want to limit our study on lowering the diffusion delay when considering large m-sequences. More precisely, our aim in this section is double: we want to propose LFSRs that produce large m-sequences with an efficient implementation and with a low diffusion delay.

Let us explain now why it is important in cryptographic context to lower diffusion delay. This criterion aims at evaluating the speed needed to completely spread a difference into the automaton. More precisely, when considering an LFSM of size n with a diffusion delay δ. Replacing the content of a cell $m_i^{(t)}$ by $m_i^{(t)} + 1$ may influence any cell m_j with $0 \leq j < n$ after δ clocks. It could also be expressed in terms of correlation: after δ clocks, the behavior of any cell is correlated with any other. More precisely, consider the two following sequences: the first sequence $\alpha = (a_1, \cdots, a_N)$ is a binary sequence of the states of the content of the register of an LFSR initialized with an n-bit word a_1 (i.e. each element a_i of α is the content at time i of the LFSR and is n-bit long). The second sequence of same length N, $\beta = (b_1, \cdots, b_N)$, is constructed in the same way with an initialization b_1 that differ from a_1 on a single bit position. Then, $C(\alpha, \beta)$ is lowered by the LFSR with the smaller diffusion delay for small values of N (we have compared the results obtained for three LFSRs of length $n = 12$ bits (a Galois one, a Fibonacci one and a Ring one) and correlation values until $N = 256$). Note that the effect of a small diffusion delay could only be observed for small values of N because after more clocks the influence of each modified bit is complete whatever the value of the diffusion delay of the considered LFSR.

For example, considering Galois, Fibonacci LFSRs and Cellular automata of size n, the associated diffusion delay is $n - 1$ because the cells on each side m_0 and m_{n-1} require $n - 1$ clocks to mix together. In the other hand, Ring LFSRs allow to lower this parameter as its associated graph is closer to a random graph, and as the expected value of the diameter of a random graph with n vertices is $\sqrt{n}$. Ring LFSRs achieve a better diffusion delay. However, in practice, this value is an average that could not be always reached especially because we also focus our design choices on Ring LFSRs with sparse transition matrix, i.e., we consider graphs with few edges.

This diffusion delay criterion may be important for cryptographic purpose where small differences in keys or in messages are required to have a large impact. It may also be useful to lower the dimension gap for Pseudo Random Generators as presented in [35], [30]. Hence, the dimension gap lowers when an RNG outputs uniformly distributed point in a given sample space.

Moreover, this diffusion delay criterion could also be important, in stream cipher design, to determine the number of clocks required by the so called initialization phase and to speed up this step. Indeed, a stream cipher is composed of two phases: an initialization phase where no bit are output and a generation phase where bits are output. The initialization phase aims at mixing together the key bits and the IV bits. Thus, a lower diffusion delay allows to speed up this mix in terms of number of clocks. For example, the F-FCSR v3 stream cipher proposed in [36] based on a ring FCSR with a diffusion delay equal to d has an initialization phase with only $d+4$ clocks for mixing purpose whereas the previous version of the F-FCSR family (F-FCSR v2) is based on a Galois FCSR and thus requires $n + 4$ clocks in the initialization step where n is the length of the considered FCSR. Thus, as $d < n$, a ring FCSR with a "good" (i.e. low) diffusion delay allows to improve the general throughput of the stream cipher by speeding up the initialization step.

As previously suggested by the example concerning FCSRs, because the diffusion delay criterion introduced in this section is essentially linked with the graph of the automaton whatever the considered graph, then the diffusion delay criterion could be applied for all possible automata: LFSRs, NLFSRs or FCSRs. For example, the FCSR used in the stream cipher F-FCSR v3 is a ring FCSR which has replaced a classical Galois FCSR. This modification leads to halve the number of required clocks during the initialization step and to completely discard the attack of Hell and Johannson [37] against F-FCSR v2 due to a better internal diffusion delay.

B. Efficient hardware design

We show in this subsection how to achieve good hardware design and we first introduce the constraints required to

achieve such a design:

- Critical path length: The shorter longest path must be as short as possible to raise frequency.
- Fan-out: A given signal should drive minimum gate number as exposed in [14].
- Cost: The number of logic gates must be as small as possible to lower consumption.

We focus on these parameters because lowering these values allows to increase the frequency of the automata, consequently it allows to increase the throughput.

1) Previous works: Previous works have been done to lower those parameters. For example, in [38] the authors proposed top-bottom LFSR: a Ring LFSR divided in two parts: a Fibonacci part and a Galois part corresponding with a transition matrix of the form:

$$A = \begin{pmatrix} g_1 & 1 & & & & & & \\ g_2 & & 1 & & & & (0) & \\ \vdots & & & \ddots & & & & \\ g_{i-1} & & & & \ddots & & & \\ & & & & & \ddots & & \\ & & (0) & & & & \ddots & \\ & & & & & & & 1 \\ 1 & & & & & f_i & f_{i+1} & \dots & f_n \end{pmatrix}$$

This approach is a trade-off between Galois and Fibonacci LFSRs. In particular, given a polynomial, there exists a top-bottom LFSR with this connection polynomial. The critical path length, the fan-out and the cost may thus be an average between the Galois and the Fibonacci cases. But this construction also carries the disadvantages of both cases, for example a slow diffusion delay.

In [17], the authors proposed a method that constructs, from a given LFSR, a similar LFSR with a lower critical path length and a lower fan-out. To do so, they modify step by step the transition matrix of the original LFSR using left and right shifts without modifying the corresponding value of the connection polynomial. For a given connection polynomial, those constructions lead to implementations with a critical path of length at most 2, a fan-out of at most 3 and a constant cost when starting the algorithm using a Galois LFSR. More precisely, their method behaves well on polynomials with uniformly distributed coefficients, i.e., polynomials with the same separation between any two consecutive non-zero coefficients. They give as an example the polynomial $X^{72} + X^{64} + X^{55} + X^{45} + X^{37} + X^{27} + X^{18} + X^9 + 1$, compared to $X^{72}+X^{49}+X^6+X^5+X^4+X^3+X^2+X^1+1$. In summary, their method leads to consider Ring LFSRs with transition matrix of the form

$$A = \begin{pmatrix} & 1 & & & & & & \\ & & 1 & & & & (0) & \\ & & & \ddots & & & & \\ & & & & 1 & & & \\ (0) & & & & h_1 & 1 & & \\ & & & \iddots & h_2 & & \ddots & \\ & & h_{n-4} & \iddots & & & & \ddots \\ & h_{n-2} & h_{n-3} & & & & & 1 \\ 1 & h_{n-1} & & & (0) & & & \end{pmatrix}$$

for the connection polynomial $X^n+h_{n-1}X^{n-1}+\cdots+h_1X+1$ and n odd (the form is similar for n even).

The authors also give a generic method (using two other elementary transformations called SDL and SDR that preserve the connection polynomial) to lower the hardware cost of an LFSR. To reach an LFSR with a better cost, the authors must apply their method step by step until a x-or operation is reached using their algorithm. The point of view taken in this article is thus from a given connection polynomial and a given transition matrix to reach a better form of the transition matrix (and thus a better hardware implementation) keeping the same connection polynomial. The proposed methods are based on looking at similar LFSRs. However, from a given LFSR, all the possible similar LFSRs could not be reached using their algorithms. The corresponding diffusion delay of this kind of LFSRs is about $n/2$. We show in the different examples given in this Section that we could reach a better diffusion delay jointly with a more compact implementation.

2) Our approach: Moreover, in most of the applications, the designer does not care about which connection polynomial is chosen for the LFSR but only needs to know that the connection polynomial is primitive. This is the core of our approach and of our proposal where we randomly pick transition matrices with desired properties (that could be application-dependent) and a posteriori verify if the obtained connection polynomial is primitive or not. To do so, we first need to express the previous required constraints relying on the transition matrix of a Ring LFSR. Table III sums up those constraints using the following notations: denote by $\mathcal{L}$ a Ring LFSR of length n with transition matrix A. We compute its connection polynomial $Q(X)$ and consider the associated Galois LFSR $\mathcal{L}_G$ and Fibonacci LFSR $\mathcal{L}_F$. We denote by $\mathrm{col}_0, \dots, \mathrm{col}_{n-1}$ the columns of A and $\mathrm{row}_0, \dots, \mathrm{row}_{n-1}$ its rows. We note $w := w_H(Q(X))$. All the presented constraints are taken into account in our approach in order to reach an LFSM that satisfies all the requirements.

Galois LFSRs are optimal for the critical path, while Fibonacci LFSRs are optimal for the fan-out. A Ring LFSR can be built to reach these two values. More precisely a Ring LFSR with a Hamming weight of at most 2 for its columns and its rows have an optimal critical path and an optimal fan-out with a good diffusion delay as summed up in Table III.

However, we do not have an algorithm that construct an LFSR with a given connection polynomial, we just can pick random transition matrix with good properties. Hence, as we

	Galois	Fibonacci	Cellular automaton	Ring LFSR	LFSR of [17]
Critical path	1	$\lceil \log_2(w-1) \rceil$	2	$\max \lceil \log_2(w_H(\mathrm{row}_i)) \rceil$	2
Fan-out	$w-1$	2	3	$\max w_H(\mathrm{col}_i)$	3
Cost	$w-2$	$w-2$	n	$w_H(T)-n$	$w-2$
Diffusion delay	$n-1$	$n-1$	$n-1$	$\leq n-1$	$n/2$

TABLE III
CRITICAL PATH, FAN-OUT, COST AND DIFFUSION DELAY OF GALOIS LFSRS, FIBONACCI LFSRS, CELLULAR AUTOMATA, GENERIC RING LFSRS AND CONSTRUCTION PROPOSED IN [17].

allow the connection to be freely chosen, the constructed matrices do not present any special form allowing to compute efficiently the connection polynomial. Moreover, when considering LFSMs in practice, the constraint on the connection polynomial is simply to be primitive, not to have a particular value.

Algorithm 1: Algorithm to pick randomly a Ring LFSR with a good hardware design.

Require: n the length of the Ring LFSR to seek. $f \leq n$ the number of feedbacks to place.
Ensure: A transition matrix A with a critical path of length 1, a fan-out of 2 and a cost of f logic gates and such that its connection polynomial is primitive of degree n.
repeat
  $A \leftarrow (a_{i,j})_{0 \leq i,j < n}$ with $a_{i,j} = \begin{cases} 1 \text{ if } j \equiv i+1 \bmod n \\ 0 \text{ otherwise} \end{cases}$
  while $w_H(A) < n + f$ **do**
    $(i,j) \leftarrow Random([0,n] \times [0,n])$
    if $w_H(\mathrm{row}_i) = 1$ AND $w_H(\mathrm{col}_j) = 1$ **then**
      $a_{i,j} \leftarrow 1$
    end if
  end while
  $Q(X) \leftarrow \det(I - XA)$
until $Q(X)$ is primitive
return A

Algorithm 1 picks random feedbacks positions and computes the associated connection polynomial. This algorithm is probabilistic. We expect picking a random matrix of size n and computing its connection polynomial is equivalent to pick a random polynomial of degree n. More precisely we know that the connection polynomial as its constant coefficient and its greatest coefficient equal to 1, so the number of possibly constructed polynomials is 2^{n-2}. The number of primitive polynomials of degree n over $\mathbb{F}_2$ is $\frac{\varphi(2^n-1)}{n}$ where φ is the Euler function. We expect Algorithm 1 to be successful after $\frac{2^{n-2}}{\varphi(2^n-1)/n}$ tries as presented in Fig. 13.

The time complexity of this algorithm is driven by the time it takes to compute $\det(I - XA)$ which is roughly $\mathcal{O}(n^3)$.

For a hardware oriented LFSM, each feedback can be freely placed. Using this property we can lower the complexity of the previous algorithm using intermediate computations done using the cofactors of the matrix A as follows:

Proposition 5.2: Given a matrix A over a ring R of size $n \times n$. Note $E_{i,j}$ the matrix with a single 1 in position i,j. Then we have $\det(A + \lambda E_{i,j}) = \det(A) + \lambda \,\mathrm{cof}_{i,j}$ where $\mathrm{cof}_{i,j}$ denotes the (i,j)-th cofactor of the matrix A.

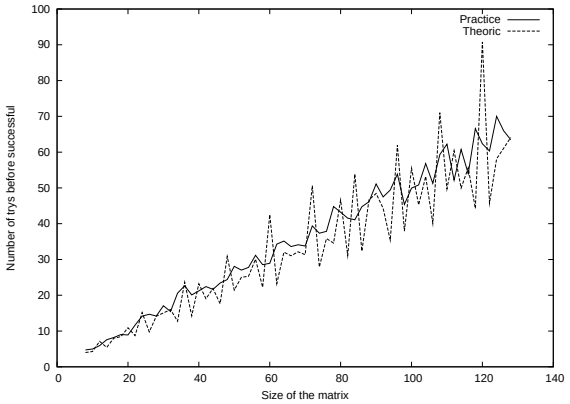


Fig. 13. Theoretic and empirical number of trials needed for Algorithm 1.

The cofactors matrix of a matrix is equal to the transposition of its adjunct matrix, which could be computed with classical inversion algorithms. Using the previous proposition, we are able to improve the complexity of our algorithm using Algorithm 2.

Algorithm 2: Algorithm to pick randomly a Ring LFSR with a good hardware design.

Require: n the length of the Ring LFSR to seek. $f \leq n$ the number of feedbacks to be placed.
Ensure: A transition matrix A with a critical path of length 1, a fan-out of 2 and a cost of f logic gates and such that its connection polynomial is primitive of degree n.
loop
  $A \leftarrow (a_{i,j})_{0 \leq i,j < n}$ with $a_{i,j} = \begin{cases} 1 \text{ if } j \equiv i+1 \bmod n \\ 0 \text{ otherwise} \end{cases}$
  while $w_H(A) < n + f - 1$ **do**
    $(i,j) \leftarrow Random([0,n] \times [0,n])$
    if $w_H(\mathrm{row}_i) = 1$ and $w_H(\mathrm{col}_j) = 1$ **then**
      $a_{i,j} \leftarrow 1$
    end if
  end while
  $C \leftarrow$ cofactors matrix of $I - XA$
  $Q_0(X) \leftarrow \det(I - XA)$
  for $0 \leq i,j < n$ **do**
    if $w_H(\mathrm{row}_i) = 1$ and $w_H(\mathrm{col}_j) = 1$ **then**
      $Q(X) \leftarrow Q_0(X) - XC_{i,j}$
      if $Q(X)$ is primitive **then**
        Break
      end if

end if
end for
end loop
return A

The complexity of this algorithm is driven by the computation of the cofactors matrix and its determinant which can be achieved by a common algorithm. Each computation of cofactors matrix costs $\mathcal{O}(n^3)$ operations. With a single cofactors matrix, we test roughly $n^2 - nf$ polynomials. So the average complexity is about $\mathcal{O}(n)$ operations.

3) Example: We give in Appendix -A an example of a hardware oriented LFSR of length 128 found using Algorithm 2. This LFSR has a primitive connection polynomial which has an Hamming weight of 65. The diffusion delay of this LFSR is only 27 whereas the corresponding diffusion delay for a Galois or a Fibonacci LFSR would be 127.

C. Efficient software and hardware design

In the previous subsection, we focus our work on an efficient algorithm to find efficient LFSRs for hardware design. In this subsection, we show how we could adapt those results for efficient software design of an LFSR and show how this design is also efficient in hardware. The main difference between hardware and software is the atomic data size. In hardware we operate on single bits, whereas in software bits are natively packed in words such that working on single bits is not natural and needs additional operations. The word size depends on the architecture of the processor: 8 bits, 16 bits, 32 bits, 64 bits or more. To benefit from this architecture we propose to use LFSRs acting on words. Let us first summarize the previous works that have been done to optimize software performances of LFSRs. Then, we introduce our construction method to build LFSRs efficient in software and in hardware.

1) Previous works: Firstly, the Generalized Feedback Shift Registers were introduced in [39] to increase the throughput. The main idea here was to parallelize w Fibonacci LFSRs. More formally, the corresponding matrix of such a construction is:

$$A = \begin{pmatrix} 0 & I_w & 0 & \dots & \dots & 0 \\ \vdots & \ddots & I_w & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ \vdots & & & \ddots & \ddots & 0 \\ 0 & \dots & \dots & \dots & 0 & I_w \\ I_w & a_{n-2}I_w & \dots & \dots & a_1 I_w & a_0 I_w \end{pmatrix}$$

where I_w represents the $w \times w$ identity matrix over $\mathbb{F}_2$ and where the a_i for i in $[0, .., n-2]$ are binary coefficients. The matrix A could be seen at bit level but also at w-bits word level, each bit of the w-bits word is in fact one bit of the internal state of one Fibonacci LFSR among the w LFSRs.

In [2], Roggeman applied the previous definition to LFSRs to obtain the Generalized Linear Feedback Shift Registers but in this case the matrix T is always defined at bit level. In 1992, Matsumoto in [40] generalized this last approach considering no more LFSR at bit level but at vector bit level (called word). This representation is called Twisted Generalized Feedback Shift Register whereas the same kind of architecture was also described in [41] and called the Mersenne Twister. In those approaches, the considered LFSRs are in Fibonacci mode seen at word level with a unique linear feedback. The corresponding matrices are of the form:

$$A = \begin{pmatrix} 0 & I_w & 0 & \dots & \dots & \dots & \dots & 0 \\ \vdots & \ddots & I_w & \ddots & & & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & & & \vdots \\ \vdots & & & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & & & \ddots & \ddots & \ddots & \vdots \\ \vdots & & & & & \ddots & \ddots & 0 \\ 0 & \dots & \dots & \dots & \dots & \dots & 0 & I_w \\ I_w & 0 & \dots & 0 & L & 0 & \dots & 0 \end{pmatrix}$$

where I_w represents the $w \times w$ identity matrix and where L is a $w \times w$ binary matrix chosen to produce m-sequences. In this case, the matrix is defined over $\mathbb{F}_2$ but could also be seen at w-bits word level. This is the first generalization of LFSRs specially designed for software applications due to the word oriented structure.

The last generalization was introduced in 1995 in [42] with the Multiple-Recursive Matrix Method and used in the Xorshift Generators described in [43] and well studied in [30]. In this case, the used LFSRs are in Fibonacci mode with several linear feedbacks. The matrix representation is:

$$A = \begin{pmatrix} 0 & I_w & 0 & \dots & \dots & 0 \\ \vdots & \ddots & I_w & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ \vdots & & & \ddots & \ddots & 0 \\ 0 & \dots & \dots & \dots & 0 & I_w \\ A_r & A_{r-1} & \dots & \dots & A_2 & A_1 \end{pmatrix}$$

where I_w is the identity matrix and where the matrices A_i are software efficient transformations such as right or left shifts at word level or word rotation. The main advantage of this representation is its word-oriented software efficiency but it also preserves all the good LFSRs properties if the underlying polynomial is primitive. Moreover, using the special form of the transition matrix, the connection polynomial is efficiently computed with the formula $P(X) = \det\left(I + \sum_{j=1}^{r} X^j A_j\right)$.

A particular case of the Multiple-Recursive Matrix Method is studied in [44]. The authors proposed to consider matrices A_i of the form $a_i \cdot T$ where T is a square matrix of size w, and a_i are scalar elements. In this case, an algorithm to construct LFSMs with primitive polynomials is given. This paper was the first to introduce efficient word-oriented LFSRs, thus solving the challenge proposed by Bart Preneel in [45].

An other way to construct software oriented LFSRs is to consider LFSRs over $\mathbb{F}_{2^w}$ as done in [7], [24]. The SNOW LFSR is given in Appendix -B. This interpretation allows to use table-lookup optimization and gives good results. Those automata could be interpreted as linear automata over $\mathbb{F}_2$

because of the mapping $\mathbb{F}_{2^w} \rightarrow (\mathbb{F}_2)^w$. In particular, they can be consider as a special case of our proposal.

2) Our proposal for building LFSRs efficient in software and in hardware: As for the hardware case our approach focuses on the construction of a software oriented transition matrix. To do so, we use transition matrices defined by block. In the next algorithm, A defines a block matrix, i.e., A is taken in $\mathcal{M}_{n/k}(\mathcal{M}_k(\mathbb{F}_2))$ for a matrix of size n divided in blocks of size k over $\mathbb{F}_2$. When an LFSR is being defined by block, we call it a word-LFSR.

Moreover we use the right and left shift operations (denoted $\gg$ and $\ll$) which are fast and implemented at word level. Given a word size k we define the matrix L of left shift as the matrix $k \times k$ with ones on its overdiagonal and zeros elsewhere. Similarly, the matrix R of right shift is defined as the matrix $k \times k$ with ones on its sub-diagonal and zeros elsewhere, such that we have:

$$\begin{aligned} L \cdot (x_0, x_1, \ldots, x_{k-1})^t &= (x_1, \ldots, x_{k-1}, 0)^t \\ R \cdot (x_0, x_1, \ldots, x_{k-1})^t &= (0, x_0, x_1, \ldots, x_{k-2})^t \end{aligned}$$

Remark that LFSRs over $\mathbb{F}_{2^w}$ can be expressed as word-LFSRs where used operations are multiplications on $\mathbb{F}_{2^w}$ seen as a space vector over $\mathbb{F}_2$, i.e., there exists a bijection between $\mathbb{F}_{2^w}$ and $(\mathbb{F}_2)^w$.

According to the previous discussion we propose Algorithm 3 to build efficient software LFSRs.

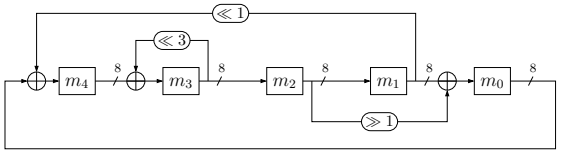
Algorithm 3: Algorithm to pick randomly an LFSR with a good software design.

Require: k the word size. n the length of the LFSR to seek with $k|n$. $f \leq n/k$ the number of word-feedbacks to place.
Ensure: A transition matrix A define by block with a cost of f shift and xor operations and such that its connection polynomial is primitive of degree n.
repeat
$A \leftarrow (a_{i,j})_{0 \leq i,j < n/k}$
with $a_{i,j} = \begin{cases} I_k \text{ if } j \equiv i+1 \bmod n/k \\ 0 \text{ otherwise} \end{cases}$
$From \leftarrow Random([0, n/k]^f)$
$To \leftarrow Random([0, n/k]^f)$
$Shift \leftarrow Random\left(([-k/2, k/2] \setminus \{0\})^f\right)$
for $l \leftarrow 0$ to $f-1$ **do**
$a_{To[l],From[l]} \leftarrow a_{To[l],From[l]} + \begin{cases} L^{Shift[l]} \text{ if } Shift[l] > 0 \\ R^{-Shift[l]} \text{ otherwise} \end{cases}$
end for
$Q(X) \leftarrow \det(I - XA)$
until $Q(X)$ is primitive
return A

This algorithm picks random word-feedbacks positions and shift values, and computes the associated connection polynomial. The complexity of this algorithm is about the same than Algorithm 1 because we have not been able to use the block structure of the matrix to lower the determinant computation complexity.

$$A = \begin{pmatrix} & I_8 & R^1 & & \\ & & I_8 & & \\ & & & I_8 & \\ & & & L^3 & I_8 \\ I_8 & L^1 & & & \end{pmatrix}$$

(a) Transition matrix



(b) Representation

Fig. 14. An LFSR with efficient software design.

3) Example: We give in Figure 14 an example of an LFSR with an efficient software design with $n = 40$ and $k = 8$ and a primitive connection polynomial. The corresponding hardware implementation of this LFSR is also very good due to its intrinsic structure (a fan out of 2, a critical path of length 1 and a cost of 19 adders) and because it fulfills the requirements of Algorithm 2. The diffusion delay of this LFSR is 27.

Let us now also compare a word oriented LFSR picked using our algorithm to the SNOW2.0 LFSR defined in [7]. The two LFSRs are respectively described in Appendix -B and in Appendix -C.

These two LFSRs output m-sequences of degree 512. We compare the diffusion delay and the throughput in software for those two LFSRs:

- The diffusion delay of the SNOW LFSR is 49 compared to 33 for our LFSR.
- The cost of one clock is 8 cycles for the SNOW LFSR using the sliding window implementation as proposed in [7] (this technique could be only applied for a Fibonacci LFSR). The cost for this LFSR implemented using classical implementation is 20 cycles. The cost for our LFSR is 33 cycles.

As presented the diffusion delay is better for our LFSR. However, the cost of one clock is higher in our case. This is due to the fact that the SNOW LFSR is sparse (three feedbacks) while ours has 8 feedbacks. Moreover, the computations are made using precomputed tables which leads to a better cost. However, the hardware implementation of our own LFSR has a really low cost (it fulfills the hardware design criteria we require in the previous section: critical path of length 1, fan-out of 2) whereas the SNOW2.0 LFSR could not be efficiently implemented in hardware due to the precomputed tables.

VI. Conclusion

In this paper, we have shown how to link together matrix representations and polynomial representations for efficient LFSMs, LFSRs and windmill generators constructions. Those new representations lead to efficient implementations both in software and in hardware. We have compared new Ring LFSR constructions with LFSRs used in several stream ciphers and

we have shown that Ring LFSRs have always a better diffusion delay with better hardware performances and good software performances.

In further works, we aim at more precisely looking at the case of an LFSM with ℓ output bits to give equivalent and general representations. We also want to generalize those new results to Finite State Machines that are no more linear. The same kind of generalization could be efficiently applied to Feedback with Carry Shift Registers (FCSRs) or to Algebraic Feedback Shift Registers (AFSRs).

To sum up the results given on diffusion, we have proposed two algorithms one for hardware purpose, one for software purpose that allow to build efficient LFSRs with a low diffusion delay and good implementation criteria. Moreover, building an LFSR using Algorithm 3 leads to an LFSR with good cryptographic properties with an efficient implementation both in software and in hardware.

A. Example of a Ring LFSR of size 128 bits

We describe a Ring LFSR of size 128 bits. The transition matrix $A = (a_{i,j})$ is given by:

$$\begin{cases} a_{i,i+1} = 1 \text{ for all } 0 \leq i < 127 \\ a_{127,0} = 1 \\ a_{i,j} = 1 \text{ for } (i,j) \in \mathcal{F} \end{cases}$$

where $\mathcal{F}$ is the set:

$$\left\{ \begin{array}{cccc} (4,78), & (5,19), & (8,44), & (9,106), \\ (10,70), & (12,14), & (14,115), & (15,55), \\ (17,82), & (21,64), & (22,12), & (25,127), \\ (27,107), & (28,112), & (31,59), & (34,111), \\ (35,48), & (37,36), & (38,23), & (39,88), \\ (43,37), & (44,26), & (46,60), & (47,100), \\ (49,24), & (50,25), & (51,2), & (51,27), \\ (55,124), & (57,113), & (59,71), & (61,29), \\ (69,123), & (72,52), & (73,118), & (77,46), \\ (80,74), & (81,83), & (83,98), & (87,53), \\ (88,73), & (91,47), & (93,10), & (94,21), \\ (95,93), & (97,13), & (98,117), & (99,50), \\ (100,3), & (101,104), & (104,1), & (105,114), \\ (106,108), & (107,105), & (109,4), & (111,28), \\ (112,68), & (113,42), & (114,31), & (119,18), \\ (120,49), & (121,32), & (123,94), & (124,6) \end{array} \right\}$$

This LFSR has a primitive connection polynomial. It has a cost of 64 adders, a fan-out equal to 2 and a critical path of 1, and a diffusion delay of 27.

B. Description of the LFSR in SNOW 2.0 over $\mathbb{F}_2$

We give here a description of the LFSR used in SNOW 2.0 [7] seen as a LFSR over $\mathbb{F}_2$. This LFSR is a Fibonacci LFSR over $\mathbb{F}_{2^{32}}$. The field $\mathbb{F}_{2^{32}}$ is defined as an extension of $\mathbb{F}_{2^8}$ to allow an efficient implementation and to prevent the guess-and-determine attack presented in [46].

The implementation is based upon the multiplication by $\alpha \in \mathbb{F}_{2^{32}}$ (the primitive polynomial of SN0W 2.0 is given by $\pi(X) = \alpha X^{16} + X^{14} + \alpha^{-1} X^5 + 1$ in $\mathbb{F}_{2^{32}}[X]$) where α is a root of $X^4 + \beta^{23} X^3 + \beta^{245} X^2 + \beta^{239}$ in $\mathbb{F}_{2^8}[X]$ and where β is a root of $x^8 + x^7 + x^5 + x^3 + 1$ in $\mathbb{F}_2[X]$. We denote M_α the matrix of this linear application seen over $\mathbb{F}_2^{32}$:

$$M_\alpha = \left(\begin{array}{ccc|c|c|c|c} 0 & 0 & 0 & & & & \\ I_8 & & (0) & V_0 & V_1 & \dots & V_7 \\ & I_8 & & & & & \\ (0) & & I_8 & & & & \end{array} \right)$$

where

$$\begin{cases} V_0 = {}^t(\texttt{0xE19FCF13}) \\ V_1 = {}^t(\texttt{0x6B973726}) \\ V_2 = {}^t(\texttt{0xD6876E4C}) \\ V_3 = {}^t(\texttt{0x05A7DC98}) \\ V_4 = {}^t(\texttt{0x0AE71199}) \\ V_5 = {}^t(\texttt{0x1467229B}) \\ V_6 = {}^t(\texttt{0x28CE449F}) \\ V_7 = {}^t(\texttt{0x50358897}) \end{cases}$$

Then the transition matrix of the LFSR of SNOW2.0 is presented in Figure 15.

C. Example of a word-oriented LFSR of size 512 bits

We give in Figure 16 a description of a word-oriented LFSR of length 512 with words of 32 bits. The grid in the matrix is drawn for readability.

ACKNOWLEDGMENT

The authors would like to thank the anonymous referees and Cédric Lauradoux for their helpful comments, remarks and suggestions.

REFERENCES

[1] S. W. Golomb, *Shift Register Sequences*. Aegen Park Press, 1981.

[2] Y. Roggeman, "Varying Feedback Shift Registers," in *EUROCRYPT*, ser. Lecture Notes in Computer Science, vol. 434. Springer, 1989, pp. 670–679.

[3] D. Kagaris, "A similarity transform for Linear Finite State Machines," *Discrete Applied Mathematics*, vol. 154, no. 11, pp. 1570–1577, 2006.

[4] B. J. M. Smeets and W. G. Chambers, "Windmill generators: A generalization and an observation of how many there are," in *EUROCRYPT*, ser. Lecture Notes in Computer Science, vol. 330. Springer,1988, pp. 325–330.

[5] Bluetooth, "Specification of the Bluetooth system, volume 1: Core, v1.1," Bluetooth SIG, February 2001.

[6] C. Lauradoux, "Extended Windmill Polynomials," in *IEEE International Symposium on Information Theory - ISIT 2009*. Seoul, Korea: IEEE, june-july 2009, pp. 1120–1124.

[7] P. Ekdahl and T. Johansson, "A new version of the stream cipher SNOW," in *Selected Areas in Cryptography – SAC 2002*, ser. Lecture Notes in Computer Science, vol. 2295. Springer, 2002, pp. 47–61.

[8] NESSIE, "Nessie phase 1: selection of primitives," https://www.cryptonessie.org/, 2001.

[9] H. Stone, "Discrete Mathematical Structures and their Applications. Sci. Res," *Associates, Chicago*, 1973.

[10] M. Goresky and A. Klapper, "Algebraic shift register sequences," 2009, avalaible at http://cs.engr.uky.edu/~klapper/algebraic.html.

[11] K. Cattell and J. C. Muzio, "An Explicit Similarity Transform between Cellular Automata and LFSR Matrices," *Finite Fields and Their Applications*, vol. 4, no. 3, pp. 239 – 251, 1998.

[12] I. Goldberg and D. Wagner, "Architectural considerations for cryptanalytic hardware," *CS252 Report¡ http://www. cs. berkeley. edu/~ iang/isaac/hardware*, 1996.

[13] P. Leglise, F. Standaert, G. Rouvroy, and J.-J. Quisquater, "Efficient implementation of recent stream ciphers on reconfigurable hardware devices," in *26th Symposium on Information Theory in the Benelux*, 2005, pp. 261–268.

$$\begin{pmatrix}
 & I_{32} & & & & & & & & & & & & & & \\
 & & I_{32} & & & & & & & & & & & & & \\
 & & & I_{32} & & & & & & & & & & & & \\
 & & & & I_{32} & & & & & & & (0) & & & & \\
 & & & & & I_{32} & & & & & & & & & & \\
 & & & & & & I_{32} & & & & & & & & & \\
 & & & & & & & I_{32} & & & & & & & & \\
 & & & & & & & & I_{32} & & & & & & & \\
 & & & & & & & & & I_{32} & & & & & & \\
 & & & & (0) & & & & & & I_{32} & & & & & \\
 & & & & & & & & & & & I_{32} & & & & \\
 & & & & & & & & & & & & I_{32} & & & \\
 & & & & & & & & & & & & & I_{32} & & \\
 & & & & & & & & & & & & & & I_{32} & \\
M_\alpha & 0 & I_{32} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & (M_\alpha)^{-1} & 0 & 0 & 0 & 0
\end{pmatrix}$$

Fig. 15. Transition matrix of SNOW2.0

$$\left(\begin{array}{ccccc|ccc|ccc|ccc|c}
 & I_{32} & & & & & & & & & & & & & \\
 & & I_{32} & & & & & & R^{14} & & & & & & \\
\hline
 & & & I_{32} & & & & L^8 & & & & & & & \\
\hline
 & & & & & & I_{32} & & & & (0) & L^{12} & & & \\
 & & & & & & & I_{32} & & & & & & & \\
 & & & & & & L^2 & I_{32} & & & & & & & \\
\hline
 & & & & & & & & I_{32} & & & & & & \\
 & & & & & & & & & I_{32} & & & & R^{11} & \\
L^{13} & & & & & & & & & & I_{32} & & & & \\
\hline
 & & & & & (0) & & & & & & I_{32} & & & \\
 & & & & & & & & & & & & I_{32} & & \\
 & & & & & & & & & & & & & I_{32} & \\
\hline
 & & & & & & & & & & & & & I_{32} & \\
 & & & & & & & & & & & R^{13} & & & I_{32} \\
\hline
I_{32} & & R^{10} & & & & & & & & & & & &
\end{array}\right)$$

Fig. 16. Transition matrix of a word oriented LFSR

[14] A. Joux and P. Delaunay, "Galois LFSR, embedded devices and side channel weaknesses," in *INDOCRYPT*, ser. Lecture Notes in Computer Science, R. Barua and T. Lange, Eds., vol. 4329. Springer, 2006, pp. 436–451.

[15] K. Cattell and J. C. Muzio, "Analysis of One-Dimensional Linear Hybrid Cellular Automata over GF(q)," *IEEE Trans. Comput.*, vol. 45, no. 7, pp. 782–792, 1996.

[16] ——, "Synthesis of One-Dimensional Linear Hybrid Cellular Automata," *IEEE Trans. Computer-Aided Design*, vol. 15, pp. 325–335, 1996.

[17] G. Mrugalski, J. Rajski, and J. Tyszer, "Ring generators - new devices for embedded test applications," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 23, no. 9, pp. 1306–1320, 2004.

[18] C. Dufaza, "Theoretical properties of LFSRs for built-in self test," *Integration*, vol. 25, no. 1, pp. 17–35, 1998.

[19] C. G. Günther, "Parallel Generation of Recurring Sequences," in *EUROCRYPT'89*, ser. Lecture Notes in Computer Science 434, 1989, pp. 503–522.

[20] E. R. Berlekamp, *Algebraic Coding Theory (revised edition)*. Aegean Park Press, 1984.

[21] E. D. Mastrovito, "VLSI Architectures for Computation in Galois Fields," Ph.D. dissertation, Linköping University, 1991.

[22] C. Paar, "Efficient VLSI Architectures for Bit Parallel Computation in Galois Fields," Ph.D. dissertation, Universität GH Essen, 1994.

[23] J. Daemen and V. Rijmen, *The Design of Rijndael: AES - The Advanced Encryption Standard*. Springer, 2002.

[24] C. Berbain, O. Billet, A. Canteaut, N. Courtois, H. Gilbert, L. Goubin, A. Gouget, L. Granboulan, C. Lauradoux, M. Minier, T. Pornin, and H. Sibert, "Sosemanuk, a fast software-oriented stream cipher," in *New Stream Cipher Designs - The eSTREAM Finalists*, ser. Lecture Notes in Computer Science, 2008, vol. 4986, pp. 98–118.

[25] M. Hell, T. Johansson, A. Maximov, and W. Meier, "The Grain family of stream ciphers," in *New Stream Cipher Designs - The eSTREAM Finalists*, ser. Lecture Notes in Computer Science, 2008, vol. 4986, pp. 179–190.

[26] E. S. C. P. eSTREAM, "The current eSTREAM portfolio," eSTREAM, ECRYPT Stream Cipher Project, 2008, http://www.ecrypt.eu.org/stream.

[27] E. Filiol and C. Fontaine, "A new ultrafast stream cipher design: Cos ciphers," in *Cryptography and Coding 2001, Proceedings*, ser. Lecture Notes in Computer Science, B. Honary, Ed., vol. 2260. Springer, 2001, pp. 85–98.

[28] J.-P. Aumasson, L. Henzen, W. Meier, and M. Naya-Plasencia, "Quark: A lightweight hash," in *CHES 2010, Proceedings*, ser. Lecture Notes in Computer Science, vol. 6225. Springer, 2010, pp. 1–15.

[29] C. Shannon, "Communication theory of secrecy systems," *Bell System Technical Journal, Vol 28, pp. 656-715*, October 1949.

[30] F. Panneton and P. L'Ecuyer, "On the Xorshift Random Number Generators," *ACM Trans. Model. Comput. Simul.*, vol. 15, no. 4, pp. 346–361,

2005.

[31] H. Feistel, “Cryptography and computer privacy,” *j-SCI-AMER*, vol. 228, no. 5, pp. 15–23, May 1973.

[32] A. F. Webster and S. E. Tavares, “On the design of S-Boxes,” in *CRYPTO’85*, ser. Lecture Notes in Computer Science, vol. 218. Springer, 1985, pp. 523–534.

[33] S. W. Golomb and G. Gong, *Signal Design for Good Correlation: For Wireless Communication, Cryptography, and Radar*. New York, NY, USA: Cambridge University Press, 2004.

[34] P. Flajolet and A. M. Odlyzko, “Random mapping statistics,” in *EUROCRYPT’89*, ser. Lecture Notes in Computer Science 434, 1989, pp. 329–354.

[35] P. L’Ecuyer, “Maximally equidistributed combined Tausworthe generators,” *Math. Comput.*, vol. 65, no. 213, pp. 203–213, 1996.

[36] F. Arnault, T. P. Berger, C. Lauradoux, M. Minier, and B. Pousse, “A new approach for FCSRs,” in *Selected Areas in Cryptography, SAC 2009, Revised Selected Papers*, ser. Lecture Notes in Computer Science, vol. 5867, 2009, pp. 433–448.

[37] M. Hell and T. Johansson, “Breaking the F-FCSR-H stream cipher in real time,” in *Advances in Cryptology - ASIACRYPT 2008*, ser. Lecture Notes in Computer Science, vol. 5350, 2008, pp. 557–569.

[38] L.-T. Wang and E. J. McCluskey, “Hybrid designs generating maximum-length sequences,” *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 7, no. 1, pp. 91–99, 1988.

[39] T. G. Lewis and W. H. Payne, “Generalized Feedback Shift Register pseudorandom number algorithm,” *J. ACM*, vol. 20, no. 3, pp. 456–468, 1973.

[40] M. Matsumoto and Y. Kurita, “Twisted GFSR generators,” *ACM Trans. Model. Comput. Simul.*, vol. 2, no. 3, pp. 179–194, 1992.

[41] M. Matsumoto and T. Nishimura, “Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator,” *ACM Trans. Model. Comput. Simul.*, vol. 8, no. 1, pp. 3–30, 1998.

[42] H. Niederreiter, “The multiple-recursive matrix method for pseudorandom number generation,” *Finite Fields Appl.*, vol. 1, no. 1, pp. 3–30, 1995.

[43] G. Marsaglia, “Xorshift RNGs,” *Journal of Statistical Software*, vol. 8, no. 14, pp. 1–6, 2003. [Online]. Available: http://www.jstatsoft.org/v08/i14;http://www.jstatsoft.org/v08/i14/xorshift.pdf

[44] B. Tsaban and U. Vishne, “Efficient Linear Feedback Shift Registers with maximal period,” *Finite Fields and Their Applications*, vol. 8, p. 256267, 2002.

[45] B. Preneel, “FSE’94 - introduction,” in *FSE*, 1994, pp. 1–5.

[46] P. Hawkes and G. G. Rose, “Guess-and-determine attacks on SNOW,” in *Selected Areas in Cryptography, SAC 2002, Revised Papers*, ser. Lecture Notes in Computer Science, vol. 2595, 2002, pp. 37–46.

François Arnault received the Ph.D. degree in mathematics in 1993, from the University of Poitiers, France. In 1994, he joined the University of Limoges, France, as an Assistant Professor, in a team working on Arithmetic, Coding-Theory and Cryptography. His research interests include Cryptography, Algorithmic Number Theory, Primality.

Thierry P. Berger received the Ph.D. degree and the French Habilitation (Mathematics) from the University of Limoges, France. From 1992, he has been with the University of Limoges. He is currently Professor in the Department of Mathematics. He is the scientific head of the Coding and Cryptography group of this department. His research interests include finite algebra, automorphism group of codes, links between coding and cryptography, stream cipher and pseudorandom generators.

Marine Minier received the Ph.D. degree in 2002, from the University of Limoges. In 2005, she joined the INSA de Lyon (Institut National des Sciences Appliquées), as an Assistant Professor, in the CITI laboratory, a team working in telecommunications. Her research interests include Symetric Key Cryptography, Security in Sensor Networks and in Ambient Networks.

Benjamin Pousse received the Ph.D. degree in 2010, from the University of Limoges. His research interests include Symetric Key Cryptography, pseudo-random generators, finite state machines.

Distinguishers for Ciphers and Known Key Attack against Rijndael with Large Blocks

Marine Minier[1], Raphael C.-W. Phan[2,⋆], and Benjamin Pousse[3]

[1] CITI Laboratory – INSA de Lyon
21 Avenue Jean Capelle, 69621 Villeurbanne Cedex – France
marine.minier@insa-lyon.fr
[2] Electronic and Electrical Engineering, Loughborough University
LE11 3TU Leicestershire – UK
R.Phan@lboro.ac.uk
[3] XLIM (UMR CNRS 6172), Université de Limoges
23 avenue Albert Thomas, F-87060 Limoges Cedex – France
benjamin.pousse@unilim.fr

Abstract. Knudsen and Rijmen introduced the notion of known-key distinguishers in an effort to view block cipher security from an alternative perspective e.g. a block cipher viewed as a primitive underlying some other cryptographic construction such as a hash function; and applied this new concept to construct a 7-round distinguisher for the AES and a 7-round Feistel cipher. In this paper, we give a natural formalization to capture this notion, and present new distinguishers that we then use to construct known-key distinguishers for Rijndael with Large Blocks up to 7 and 8 rounds.

Keywords: Block ciphers, cryptanalysis, known-key distinguishers, Rijndael.

1 Introduction

Rijndael-b is an SPN block cipher designed by Joan Daemen and Vincent Rijmen [4]. It has been chosen as the new advanced encryption standard by the NIST [6] with a 128-bit block size and a variable key length, which can be set to 128, 192 or 256 bits. In its full version, the block lengths b and the key lengths Nk can range from 128 up to 256 bits in steps of 32 bits, as detailed in [4] and in [9]. There are 25 instances of Rijndael. The number of rounds Nr depends on the text size b and on the key size Nk and varies between 10 and 14 (see Table 1 for partial details). For all the versions, the current block at the input of the round r is represented by a $4 \times t$ with $t = (b/32)$ matrix of bytes $A^{(r)}$:

⋆ Part of this work done while the author was with the Laboratoire de sécurité et de cryptographie (LASEC), EPFL, Switzerland.

B. Preneel (Ed.): AFRICACRYPT 2009, LNCS 5580, pp. 60–76, 2009.

$$A^{(r)} = \begin{pmatrix} a_{0,0}^{(r)} & a_{0,1}^{(r)} & \cdots & a_{0,t}^{(r)} \\ a_{1,0}^{(r)} & a_{1,1}^{(r)} & \cdots & a_{1,t}^{(r)} \\ a_{2,0}^{(r)} & a_{2,1}^{(r)} & \cdots & a_{2,t}^{(r)} \\ a_{3,0}^{(r)} & a_{3,1}^{(r)} & \cdots & a_{3,t}^{(r)} \end{pmatrix}$$

The round function, repeated $Nr - 1$ times, involves four elementary mappings, all linear except the first one:

- SubBytes: a bytewise transformation that applies on each byte of the current block an 8-bit to 8-bit non linear S-box S.
- ShiftRows: a linear mapping that rotates on the left all the rows of the current matrix. the values of the shifts (given in Table 1) depend on b.
- MixColumns: a linear matrix multiplication; each column of the input matrix is multiplied by the matrix M that provides the corresponding column of the output matrix.
- AddRoundKey: an x-or between the current block and the subkey of the round r K_r.

Those $Nr - 1$ rounds are surrounded at the top by an initial key addition with the subkey K_0 and at the bottom by a final transformation composed by a call to the round function where the MixColumns operation is omitted. The key schedule derives $Nr + 1$ b-bits round keys K_0 to K_{Nr} from the master key K of variable length.

Table 1. Parameters of the Rijndael block cipher where the triplet (i, j, k) for the ShiftRows operation designated the required number of byte shifts for the second row, the third one and the fourth one

	AES	Rijndael-160	Rijndael-192	Rijndael-224	Rijndael-256
ShiftRows	(1,2,3)	(1,2,3)	(1,2,3)	(1,2,4)	(1,3,4)
Nb rounds (Nk=128)	10	11	12	13	14
Nb rounds (Nk=192)	12	12	12	13	14
Nb rounds (Nk=256)	14	14	14	14	14

The idea of exploiting distinguishers for cryptanalyzing block ciphers is well known: a key-recovery attack on block ciphers typically exploits a distinguisher [11]: a structural or statistical property exhibited by a block cipher for a randomly chosen secret key K that is not expected to occur for a randomly chosen permutation. Aside from being used subsequently in key-recovery attacks, so far it seems unclear if there are any other undesirable consequences due to distinguishers, although their existence tends to indicate some certificational weakness in ciphers.

Knudsen and Rijmen [12] recently considered block cipher distinguishers when the cipher key is known to the adversary, and suggested another exploitation of the existence of distinguishers: truncated differential distinguishers lead to

near collisions in some hash function compression functions built upon block ciphers, e.g. the Matyas-Meyer-Oseas (MMO) mode [15]. Generalizing, we can similarly say for a compression function constructed from a block cipher in any of the Preneel-Govaerts-Vandewalle (PGV) modes [16], that near collisions in the ciphertext of the underlying cipher translate to near collisions in the compression function's output chaining variable. Knudsen and Rijmen posed as an open problem if a security notion exists that can capture the kind of known-key distinguishers that they proposed, and yet which would rule out non-meaningful and contrived distinguishing attacks.

This paper takes a step to answering this question. We define a security notion to express the existence of known-key distinguishers for block ciphers in Section 2. Rather than settle for a notion that is meaningful solely when the key is known to the adversary, our notion intuitively also gives some indication on the cipher's security in the conventional unknown-key setting.

Many cryptanalyses have been proposed against Rijndael-b, the first one against all the versions of Rijndael-b is due to the algorithm designers themselves and is based upon integral properties ([2], [3], [13]) that allows to efficiently distinguish 3 Rijndael inner rounds from a random permutation. This attack has been improved by Ferguson et al. in [5] allowing to cryptanalyse an 8 rounds version of Rijndael-b with a complexity equal to 2^{204} trial encryptions and $2^{128} - 2^{119}$ plaintexts.

Following the dedicated work of [7], this paper presents new four-round integral properties of Rijndael-b and the resulting 7 and 8 rounds known key distinguishers in Section 3.

2 Notions for Cipher Distinguishers

2.1 Definitions

Consider a family of functions $F : \mathcal{K} \times \mathcal{M} \to \mathcal{R}$ where $\mathcal{K} = \{0,1\}^k$ is the set of keys of F, $\mathcal{M} = \{0,1\}^l$ is the domain of F and $\mathcal{R} = \{0,1\}^L$ is the range of F, where k, l and L are the key, input and output lengths in bits. $F_K(\mathcal{M})$ is shorthand for $F(K, \mathcal{M})$. By $K \xleftarrow{\$} \mathcal{K}$, we denote randomly selecting a string K from $\mathcal{K}$. Similar notations apply for a family of permutations $E : \mathcal{K} \times \mathcal{M} \to \mathcal{M}$ where $\mathcal{K} = \{0,1\}^k$ is the set of keys of E and $\mathcal{M} = \{0,1\}^l$ is the domain and the range of E. Let $\mathrm{Func}(\mathcal{M})$ denotes the set of all functions on $\mathcal{M}$, and $\mathrm{Perm}(\mathcal{M})$ denotes the set of all permutations on $\mathcal{M}$. Let $G \xleftarrow{\$} \mathrm{Perm}(\mathcal{M})$ denotes selecting a random permutation.

The usual security notion one requires from a block cipher is to look like a pseudo-random permutation (PRP), for the keys uniformly drawn. This notion could be formalized as follows: a PRP adversary $\mathcal{A}$ gets access to an oracle, which, on input $P \in \mathcal{M}$, either returns $E_K(P)$ for a random key $k \in \mathcal{K}$ or returns $G(P)$ for a random permutation $G \in \mathrm{Perm}(\mathcal{M})$. The goal of $\mathcal{A}$ is to

guess the type of oracle it has - by convention, $\mathcal{A}$ returns 1 if it thinks that the oracle is computing $E_K(\cdot)$. The adversary's advantage is defined by:

$$Adv_E^{PRP}(\mathcal{A}) = |\Pr\left[K \xleftarrow{\$} \mathcal{K} : A^{E_K(\cdot)} = 1\right] - \Pr\left[G \xleftarrow{\$} \mathrm{Perm}(\mathcal{M}) : A^{G(\cdot)} = 1\right]|$$

E is said PRP-secure if for any $\mathcal{A}$ attacking E with resources, the advantage $Adv_E^{PRP}(\mathcal{A})$ is negligible (denoted by ε). The above notion does not take into account the decryption access. Hence, the stronger notion of Super Pseudo-Random Permutation (SPRP): as above, the adversary $\mathcal{A}$ gets access to an oracle, but in this case, the adversary not only accesses the permutations G and E_K but also their inverses G^{-1} and E_K^{-1}:

$$\begin{aligned} Adv_E^{SPRP}(\mathcal{A}) = |\Pr&\left[K \xleftarrow{\$} \mathcal{K} : A^{E_K(\cdot),E_K^{-1}(\cdot)} = 1\right] \\ &-\Pr\left[G \xleftarrow{\$} \mathrm{Perm}(\mathcal{M}) : A^{G(\cdot),G^{-1}(\cdot)} = 1\right]| \end{aligned}$$

As done by Luby and Rackoff in [14], formal results using those notions are stated with concrete bounds. However, in the "real life", we could only say that if a distinguisher exists for a given cipher E_K, it is not a PRP or a SPRP (according the distinguisher used). Note (as done in [17]) that a (adversarial) distinguisher is a (possibly computationally unbounded) Turing machine $\mathcal{A}$ which has access to an oracle $\mathcal{O}$; with the aim to distinguish a cipher E_K from the perfect cipher G by querying the oracle with a limited number n of inputs. The oracle $\mathcal{O}$ implements either E_K (for a key randomly chosen) or G. The attacker must finally answer 0 or 1. We measure the ability to distinguish E_K from G by the advantage $Adv_E(\mathcal{A}) = |p - p^*|$ (that must be true for a large part of the key space) where p (resp. $p*$) is the probability of answering 1 when $\mathcal{O}$ implements E_K (resp. G). Note also that three main classes of distinguishers exist: the non-adaptive distinguishers class (where the n plaintext queries are pre-computed), the adaptive distinguishers class (where the plaintext queries depend on the previous ones) and the super pseudo-random distinguishers one (where the queries are chosen according the previous ones and where the oracle also gets access to inverses of E_K and G).

2.2 Notions for Distinguishers

A generic definition of an n-limited non-adaptive distinguisher is given in [10] and described in Alg. 1. One gives an oracle $\mathcal{O}$ to Algorithm 1, which implements either E_K or G with probability $\frac{1}{2}$ each. The core of the distinguisher is the acceptance region $A^{(n)}$: it defines the set of input values $\mathbf{P} = (P_1, \cdots, P_n)$ which lead to output 0 (i.e. it decides that the oracle implements E_K) or 1 (i.e. it decides that the oracle implements G). The goal of the distinguisher is thus to decide whether $\mathcal{O}$ implements E_K or G. If a particular relation R that defines the acceptance region exists linking together inputs and outputs for a sufficient number of values (this efficiently-computable relation outputs 0 if the link exists

and 1 otherwise), the advantage of the non-adaptive distinguisher $Adv_E^{NA}(\mathcal{A})$ will be non-negligible. Note also that this distinguisher must work for a large part of the key space.

Algorithm 1. An n-limited generic non-adaptive distinguisher (NA)

Parameters: a complexity n, an acceptance set $A^{(n)}$
Oracle: an oracle $\mathcal{O}$ implementing a permutation c
  Compute some messages $\mathbf{P} = (P_1, \cdots, P_n)$
  Query $\mathbf{C} = (C_1, \cdots, C_n) = c(P_1, \cdots, P_n)$ to $\mathcal{O}$
  if $\mathbf{C} \in A^{(n)}$ **then**
    Output 1
  else
    Output 0
  end if

This distinguisher is generic and includes the following cases: known plaintexts distinguisher and chosen plaintexts distinguisher. In the second case, the n inputs $\mathbf{P} = (P_1, \cdots, P_n)$ must be pre-defined according a particular filter h_1 (independent from the key). By misuse of language, we use the notation $h_1(\mathbf{P})$ to designate a plaintexts set "correctly" chosen, i.e. that verifies the requirements of h_1. The acceptance region $A^{(n)}$ could be seen as the necessary minimal number of outputs in $\mathbf{C}$ that verify a particular relation $R(h_1(\mathbf{P}), \mathbf{C})$. This relation must be independent from the key or have a high probability to happen for a large class of the keys. In this case, if a such relation R that happens with a certain probability exists between the inputs and outputs sets $h_1(\mathbf{P})$ and $\mathbf{C}$; then, the advantage of the distinguisher could be non-negligible.

To illustrate how this notion applies to the existence of distinguishers for block ciphers, consider E_K as 3-round AES. Let $h_1(\mathbf{P}) = \{P_i\}_{i=0}^{255}$ be a set of 2^8 plaintexts that in one byte each has one of 2^8 possible values, and equal in all other bytes (this defines h_1); and $\mathbf{C} = \{C_i\}_{i=0}^{255}$ denote the corresponding ciphertexts, i.e. $\mathbf{C} = E_K(\mathbf{P})$. Define C_i as a concatenation of 16 bytes i.e. $C_i = C_{i,0}||C_{i,1}||\ldots||C_{i,15}$. Define $R(h_1(\mathbf{P}), \mathbf{C})$ as $\bigoplus_{i=0}^{255} C_{i,j}$ for $j = 0 \ldots 15$ for the particular set $\mathbf{C} = E_K(h_1(\mathbf{P}))$ which outputs 1 (accept) if $\bigoplus_{i=0}^{255} C_{i,j} = 0$ for $j = 0 \ldots 15$ knowing that $\mathbf{P} = \{P_i\}_{i=0}^{255}$ and that $\bigoplus_{i=0}^{255} P_{i,j} = 0$; and outputs 0 otherwise. Thus, for the case of E_K, the probability that $R(h_1(\mathbf{P}), \mathbf{C})$ outputs 1 (accept) is 1, while for the case of a random permutation G, the probability is 2^{-l}. Hence $\mathbf{Adv}_{E_K,G}^{NA\text{-}CPA}(\mathcal{A}) = 1 - 2^{-l} >> \varepsilon$ where $NA\text{-}CPA$ means non-adaptive chosen plaintexts. And so, a distinguisher exists for 3-round AES. In fact, this is the well known 3-round integral distinguisher.

As defined in [17], the super pseudo-random distinguisher (described in Alg. 2) could be defined in a deterministic way because no upper bound on the computational capability of the distinguisher are supposed to be (the only limitation is on the number of queries to the oracle).

In answer to the question posed in [12], we now define a natural extension of the above described n-limited distinguishers, to capture the kind of distinguisher

Algorithm 2. An n-limited generic adaptive distinguisher with chosen input plaintexts or output ciphertexts

Parameters: functions $g_1, \cdots, g_n$, a set $A^{(n)}$
Oracle: an oracle $\mathcal{O}$ implementing permutations c and c^{-1}
  Select a fixed direction and message $(B_1, Z_1^0) = g_1()$ and get $Z_1^1 = c(Z_1^0)$ if $B_1 = 0$ or $Z_1^1 = c^{-1}(Z_1^0)$ otherwise
  Calculate a direction and a message $(B_2, Z_2^0) = g_2(Z_1^1)$ and get $Z_2^1 = c(Z_2^0)$ if $B_2 = 0$ or $Z_2^1 = c^{-1}(Z_2^0)$ otherwise
  ...
  Calculate a direction and a message $(B_n, Z_n^0) = g_n(Z_1^1, \cdots, Z_{n-1}^1)$ and get $Z_n^1 = c(Z_n^0)$ if $B_n = 0$ or $Z_n^1 = c^{-1}(Z_n^0)$ otherwise
  if $(Z_1^1, \cdots, Z_n^1) \in A^{(n)}$ **then**
    Output 1
  else
    Output 0
  end if

that interests us in this paper and in [12]: the non-adaptive chosen middletexts one. This is shown in Alg. 3. The oracle processes the middletexts supplied by the adversary moving in either/both directions towards plaintext and/or ciphertext ends. This notion also intuitively captures the setting of known-key attacks [12] since the oracle has the same kind of power to that of an adversary having knowledge of the key.

Algorithm 3. An n-limited generic non-adaptive chosen middletexts distinguisher (NA-CMA)

Parameters: a complexity n, an acceptance set $A^{(n)}$
Oracle: an oracle $\mathcal{O}$ implementing internal functions f_1 (resp. f_2) of permutation c that process input middletexts to the plaintext (resp. ciphertext) end
  Compute some middletexts $\mathbf{M} = (M_1, \cdots, M_n)$
  Query $\mathbf{P} = (P_1, \cdots, P_n) = (f_1(M_1), \cdots, f_1(M_n))$ and $\mathbf{C} = (C_1, \cdots, C_n) = (f_2(M_1), \cdots, f_2(M_n))$ to $\mathcal{O}$
  if $(\mathbf{P}, \mathbf{C}) \in A^{(n)}$ **then**
    Output 1
  else
    Output 0
  end if

To see how this notion properly captures the 7-round known-key distinguisher for AES proposed by Knudsen and Rijmen [12], let E_K be 7-round AES, with no MixColumns in the last round. Let $\mathbf{M} = \{M_i\}_{i=0}^{2^{56}-1}$ denote the set of 2^{56} intermediate texts at the output of round 3 of E_K, that differ in seven bytes for $j = \{0, 1, 2, 3, 5, 10, 15\}$ and which have constant values in the remaining nine bytes. Let $\mathbf{P} = \{P_i\}_{i=0}^{2^{56}-1}$ be a set of 2^{56} plaintexts corresponding to the partial

decryption[1] of $\mathbf{M}$ by 3 rounds in reverse thus $\mathbf{P}$ is the plaintext subset input to E_K. Note that $\mathbf{P}$ defined in this way by Knudsen and Rijmen is only computable provided the round keys in rounds 1 to 3 are known to the adversary, or alternatively the key K is known, or one assumes the adversary can start in the middle by choosing the middletexts. This is not a problem since it is allowed by our notion. Let $\mathbf{C} = \{C_i\}_{i=0}^{2^{56}-1}$ denote the corresponding ciphertexts, i.e. $\mathbf{C} = E_K(\mathbf{P})$. Define C_i as a concatenation of 16 bytes i.e. $C_i = C_{i,0}||C_{i,1}||\ldots||C_{i,15}$. Define $R(\mathbf{P}, \mathbf{C}) = (\bigoplus_{i=0}^{2^{56}-1} P_{i,j}, \bigoplus_{i=0}^{2^{56}-1} C_{i,j})$ for $j = 0 \ldots 15$ and which outputs 1 if $\bigoplus_{i=0}^{2^{56}-1} C_{i,j} = 0$ for $j = 0 \ldots 15$ knowing that $\mathbf{P} = \{P_i\}_{i=0}^{2^{56}-1}$ and that $\bigoplus_{i=0}^{2^{56}-1} P_{i,j} = 0$; and outputs 0 otherwise. Thus, for the case of E_K, the probability that $R(\mathbf{P}, \mathbf{C})$ outputs 1 (accept) is 1, while for the case of a random permutation G, the probability is 2^{-l}.

Hence $\mathbf{Adv}_{E_K,G}^{NA\text{-}CMA}(\mathcal{A}) = 1 - 2^{-l} >> \varepsilon$. And so, a chosen middletext (a.k.a. known-key [12]) distinguisher exists for 7-round AES.

In the same way, the notion captures the 7-round distinguisher of [12] for a particular kind of Feistel cipher whose round function has the round key exclusive-ORed to the round function input, followed by an arbitrary key-independent transformation.

With this notion, we can also intuitively define security against the existence of distinguishers in the conventional setting where the key is unknown, which can be seen as a special case of $NA\text{-}CMA$.

Note here that it is apparent in the unknown-key setting that f_1 and f_2 are public functions, since it can in no way be dependent on the key, otherwise, the relation $R(\cdot, \cdot)$ becomes impossible to compute and thus verify. We defer the more detailed discussion to subsection 2.3.

2.3 Discussion

Observe that for the conventional setting where the adversary has no knowledge of the key K, it is intuitive that the distinguishing relation R operates on publicly computable functions f_1 and f_2 of the ciphertext set, otherwise if f_1 or f_2 is dependent on K, the relation cannot be verified since K is assumed to be unknown and must be uniformally distributed. Thus, the resultant notion becomes more meaningful and rules out trivial attacks where the adversary obtains a non-negligible advantage but for which is not meaningful.

Consider if the functions f_1 and f_2 in the $NA\text{-}CMA$ notion can be dependent on K, and indeed why not since K is known to A. Then take E_K to be infinitely many rounds of the AES, i.e. the number of rounds $r >> 7$. Then an adversary could still use the 7-round known-key distinguisher described in [12] as follows: peel off any number of rounds since it knows the cipher key and thus round keys to any round, and going backwards in reverse from the ciphertext end it peels off round by round until the output of round 7, and checks that the 7-round distinguisher covering the first 7 rounds is satisfied. Thus his advantage

[1] Note that this partial decryption is computable by the adversary since it knows the block cipher key K.

$\mathbf{Adv}^{NA\text{-}CMA}_{E_K,G}(\mathcal{A})$ is trivially non-negligible, although it is clear that we gain no insight into the ciphers security nor insecurity.

Thus, considering known-key distinguishers is a stronger notion than the conventional notion of unknown-key distinguishers, and so the inexistence of known-key distinguishers implies the inexistence of unknown-key distinguishers. Furthermore, it is hoped that this context might tell something about the security margin of a cipher, i.e. if an unknown-key distinguisher exists for the cipher up to s rounds, what is the most number of rounds t of the cipher for which is covered by a distinguisher if the key is known to the adversary. For instance, a distinguisher exists for the AES in the unknown-key setting up to 4 rounds [8], while Knudsen and Rijmen showed that a distinguisher exists for AES in the known-key context up to 7 rounds. This still allows some security margin considering AES has at least 10 rounds. From this perspective, (in)existence of known-key distinguishers can be viewed on the one hand as a certificational strength/weakness of a cipher; and on the other hand one still desires to gain some insight on the (in)security of a cipher from these known-key distinguishers. One obvious aim is what can be learned about unknown-key distinguishers from these known-key distinguishers.

Moreover, the only difference between the $NA\text{-}CPA/NA\text{-}CCA$ and $NA\text{-}CMA$ settings is in the extra control afforded to the latter adversary who effectively can choose his plaintext subset based on knowledge of the key what is really appreciable in the context of a meet in the middle attack. We can hence say that existence of known-key distinguishers exhibits some potentially undesirable structural property of the cipher, but which cannot be exploited in unknown-key distinguishers for key-recovery attacks only in the fact that the adversary is expected to not be able to choose the plaintext subset corresponding to the known-key distinguisher since he does not know the key. For the case of the AES, the adversary cannot turn the 7-round known-key distinguisher of Knudsen and Rijmen into an unknown-key distinguisher because he does not know the rounds keys for rounds 1 to 3.

Interestingly, this in some way relates to the motivation behind the design of cipher key schedules that have known-roundkey security, i.e. knowledge of one or more round keys does not lead to the knowledge of other round keys. In this context, known-key distinguishers can potentially be exploited in actual key-recovery attacks. Taking the 7-round AES as an example but where its key schedule has known-roundkey security: if the adversary already knows the round keys for rounds 1 to 3, and by design he still cannot obtain the other round keys, nevertheless due to the existence of the 7-round known-key distinguisher, he can use his knowledge of round keys 1 to 3 to choose a plaintext subset that corresponds to the distinguisher, and with this distinguisher he can cover all the rounds through to round 7. This can be turned into a key-recovery attack on the round keys for rounds 4 to 7.

Relation to correlation intractability. In motivating the need for a notion for known-key distinguishers, Knudsen and Rijmen discuss the related work of Canetti et al. [1] that considered the notion of *correlation intractability*, that

when applied to the context of block ciphers where the key is known to the adversary, can be seen as follows: a correlation intractable cipher is one where there exists no binary relation R between the plaintext and ciphertext that is satisfied with non-negligible probability. Clearly, if the key is known, a relation can always be found and thus a cipher cannot be correlation intractable. Yet, this trivial case is ruled out from our $NA\text{-}CMA$ notion because of the restriction we put on R to be independent of the key. Indeed, Canetti et al.'s impossibility example cannot apply in our notion, since they directly input the key to the relation, while this is not allowed for our relation.

3 Known-Key Distinguishers for the Rijndael-*b* Block Cipher with Large Blocks

We present in this Section known key distinguishers against the Rijndael-b block cipher. Using particular new and old integral properties, the building distinguishers use really few middle-texts and have very low complexity.

In [13], L. Knudsen and D. Wagner analyze integral cryptanalysis as a dual to differential attacks particularly applicable to block ciphers with bijective components. A first-order integral cryptanalysis considers a particular collection of m words in the plaintexts and ciphertexts that differ on a particular component. The aim of this attack is thus to predict the values in the sums (i.e. the integral) of the chosen words after a certain number of rounds of encryption. The same authors also generalize this approach to higher-order integrals: the original set to consider becomes a set of m^d vectors which differ in d components and where the sum of this set is predictable after a certain number of rounds. The sum of this set is called a dth-order integral.

We first introduce and extend the consistent notations proposed in [13] for expressing word-oriented integral attacks. For a first order integral, we have:

- The symbol '$\mathcal{C}$' (for "Constant") in the ith entry, means that the values of all the ith words in the collection of texts are equal.
- The symbol '$\mathcal{A}$' (for "All") means that all words in the collection of texts are different.
- The symbol '?' means that the sum of words can not be predicted.

For a dth order integral cryptanalysis:

- The symbol '$\mathcal{A}^d$' corresponds with the components that participate in a dth-order integral, i.e. if a word can take m different values then $\mathcal{A}^d$ means that in the integral, the particular word takes all values exactly m^{d-1} times.
- The term 'A_i^d' means that in the integral the string concatenation of all words with subscript i take the m^d values exactly once.
- The symbol '$(\mathcal{A}_i^d)^k$' means that in the integral the string concatenation of all words with subscript i take the m^d values exactly k times.

3.1 Known Key Distinguisher for the AES

As mentioned in [12], we could build a 4-th order 4-round AES integral distinguisher considering that the last round does not contain a MixColumns operation. Then, and as shown in Fig. 1.a, all bytes of the ciphertexts are balanced in the 4 rounds integral. Moreover, a backward 3-round property could be built for three complete rounds as shown in Fig. 1.b.

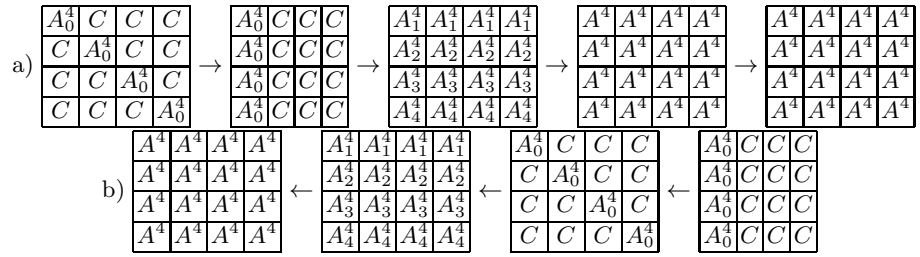


Fig. 1. a) The forward 4-round integral distinguisher with 2^{32} texts. b) A backward integral for three (full) rounds of AES with 2^{32} texts.

By applying the inside-out concatenation technique with the two previous properties, the authors of [12] could build a 7-round known key distinguisher (as shown on Fig. 2) against the AES. One chooses a structure of 2^{56} middle-texts: it has 7 active bytes whereas the other bytes are constant. We thus have 2^{24} sets of 2^{32} middletexts that represent first 2^{24} copies of the 4-round property (of Fig. 1.a) and also 2^{24} copies of the backward 3-round property (of Fig. 1.b). Then, when someone starts in the middle of the cipher, one can compute integral balanced property on both the reverse and forward directions.

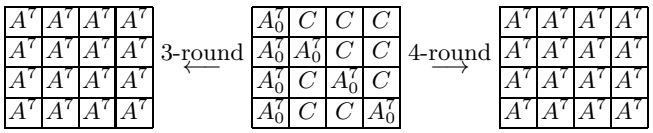


Fig. 2. The 7-round AES distinguisher with 2^{56} middle-texts. The 7th round is without MixColumns.

This known-key distinguisher simply records the frequencies in each byte of the plaintexts and ciphertexts, checks whether the values in each byte of the plaintexts and in each byte of the ciphertexts occur equally often. The time complexity is similar to the time it takes to do 2^{56} 7-round AES encryptions and the memory needed is small.

The authors of [12] introduce the k-sum problem (i.e. to find a collection of k texts $x_1, \cdots, x_k$ such as $\sum_{i=1}^{k} f(x_i) = 0$ for a given permutation f) with a

running time of $\mathcal{O}(k2^{n/(1+\log_2 k)})$ to conjecture that for a randomly chosen 128-bit permutation such a collection of texts with balanced properties could not be easily (i.e. in a reasonable computational time) found with a large probability. More precisely, the k-sum problem indicates with $n = 128$ and $k = 256$ a running time of 2^{58} operations ignoring small constants and memory. The k-sum problem is the best known approach in this case but does not give the particular balanced properties induced by the distinguisher. Thus, the authors conjecture that they have found a known-key distinguisher for AES reduced to 7 rounds using 2^{56} texts.

3.2 Rijndael-256

Thus, we have studied the same kind of properties for Rijndael-b. For the particular case of Rijndael-256, we have the two following forward and backward integral properties described in Fig. 3 and in Fig. 5. Note that the integral property of Fig. 3 could be extended by one round at the beginning using the method described in Fig. 4. This property is the one described in [7].

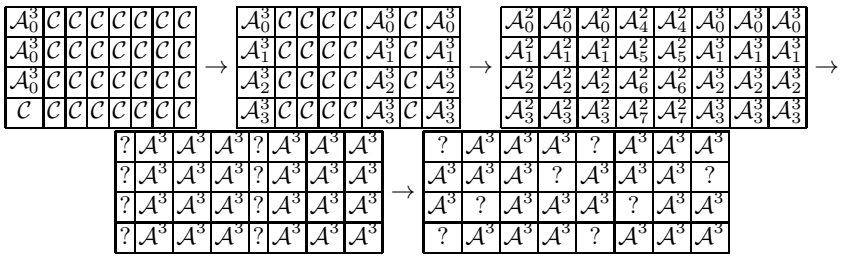


Fig. 3. 4-round 3th-order forward integral property of Rijndael-256 without the last MixColumns operation

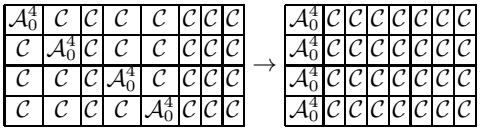


Fig. 4. Extension of a 4th order integral property by one round at the beginning for Rijndael-256

Using those two properties and the corresponding extension, we could build a 8-round known key distinguisher shown in Fig. 6. The process is exactly the same than the one described in 3.1 and the time complexity is similar to the time it takes to do 2^{40} 8-round Rijndael-256 encryptions and the memory needed is small. If we also use the k-sum problem to estimate the corresponding time to find a k-sum for a 256-bit permutation with $n = 256 - 64$ and $k = 2^{40}$, the

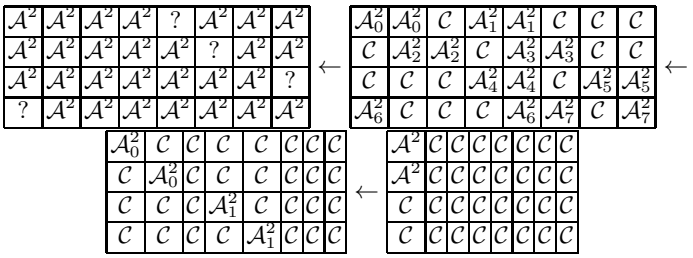


Fig. 5. The 2th-order backward 3-round integral property of Rijndael-256

corresponding complexity is around 2^{44} operations ignoring small constants and memory. Thus, we conjecture that we have found a known-key distinguisher for Rijndael-256 reduced to 8 rounds using 2^{40} middle-texts.

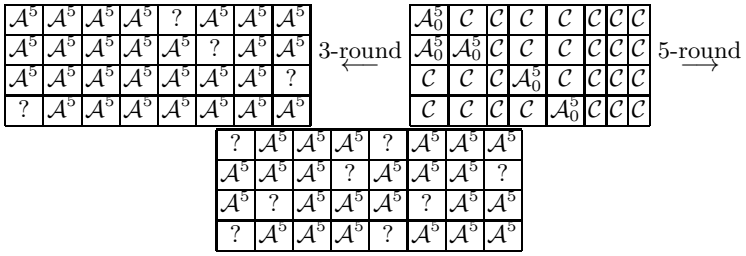


Fig. 6. The 8-round Rijndael-256 known-key distinguisher with 2^{40} middle-texts. The 8th round is without MixColumns.

3.3 Rijndael-224

Similarly, we found a 2th-order 4-round forward integral property for Rijndael-224 as shown in figure 7. We have found 42 2th-order integral properties (essentially the shifted ones). As previously done, this 2th-order four-round property could be extended by one round at the beginning using a 8th-order integral (considering that it represents 2^{48} copies of the 2th-order four-round integral). We also have found the backward 3-round 2-th order integral property for Rijndael-224 shown in Fig. 8.

Using those two properties and the corresponding extension, we could build a 8-round known key distinguisher shown in Fig. 9. The process is exactly the same than the one described in 3.1 and the time complexity is similar to the time it takes to do 2^{72} 8-round Rijndael-224 encryptions and the memory needed is small. If we also use the k-sum problem to estimate the corresponding time to find a k-sum for a 224-bit permutation with $n = 224 - 128$ and $k = 2^{72}$, the corresponding complexity is around $2^{73.8}$ operations ignoring small constants and memory. Thus, we conjecture that we have found a known-key distinguisher for Rijndael-224 reduced to 8 rounds using 2^{72} middle-texts.

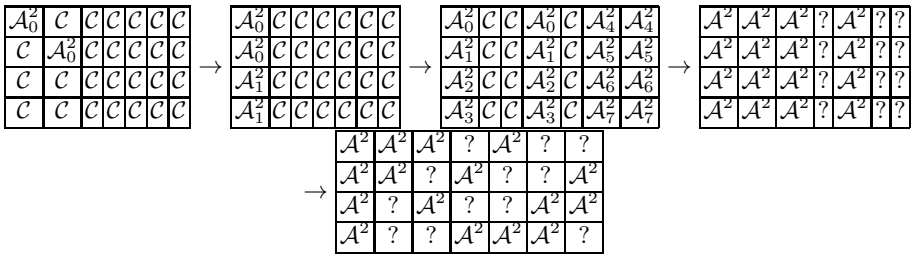


Fig. 7. Four-round 2th-order forward integral property of Rijndael-224. The last MixColumns is omitted.

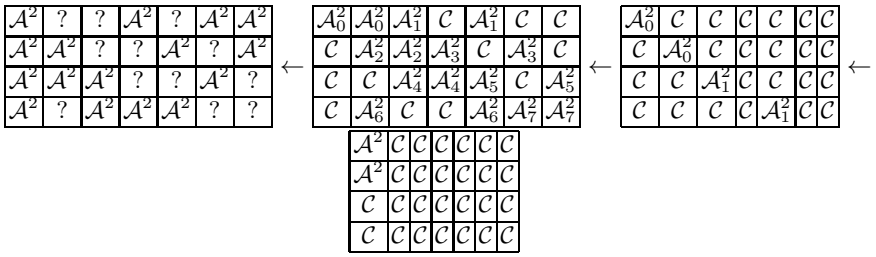


Fig. 8. The 2th-order 3-round backward integral property of Rijndael-224

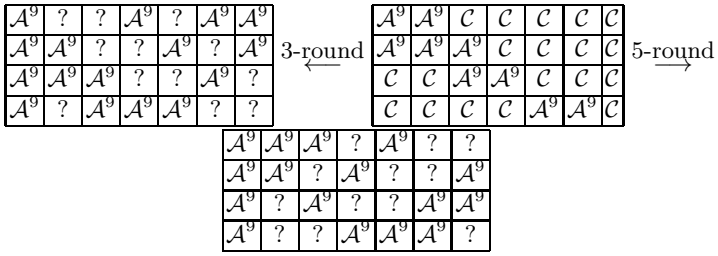


Fig. 9. The 8-round Rijndael-224 known-key distinguisher with 2^{72} middle-texts. The 8th round is without MixColumns.

3.4 Rijndael-192

In the same way, we have found a 3th-order forward 4-round integral property for Rijndael-192 as shown in Fig. 10. We have found 42 3th-order integral properties (essentially the shifted ones). We also have found the backward 2-th order integral property for Rijndael-224 shown in Fig. 11.

Using those two properties and the corresponding extension, we could build a 7-round known key distinguisher shown in Fig. 12. The process is exactly the same than the one described in 3.1 and the time complexity is similar to the time it takes to do 2^{32} 7-round Rijndael-192 encryptions and the memory needed is

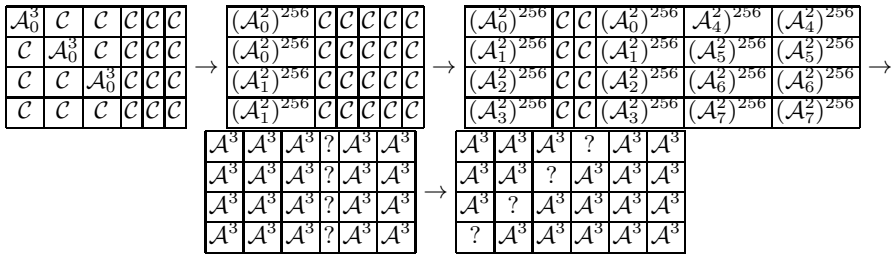


Fig. 10. The 3th-order 4-round forward integral property of Rijndael-192 (the last MixColumns is omitted)

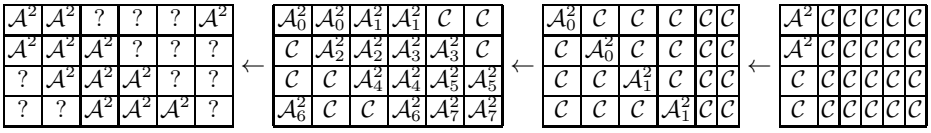


Fig. 11. 2th-order 3-round backward integral property of Rijndael-192

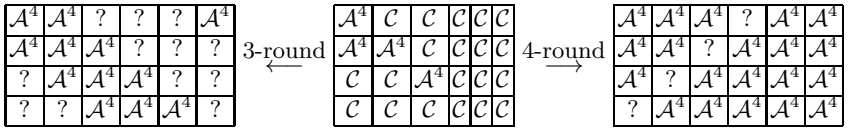


Fig. 12. The 7-round Rijndael-192 distinguisher with 2^{32} middle-texts. The 7th round is without MixColumns.

small. If we also use the k-sum problem to estimate the corresponding time to find a k-sum for a 192-bit permutation with $n = 192 - 96$ and $k = 2^{32}$, the corresponding complexity is around $2^{35.9}$ operations ignoring small constants and memory. Thus, we conjecture that we have found a known-key distinguisher for Rijndael-192 reduced to 7 rounds using 2^{32} middle-texts.

3.5 Rijndael-160

We also have found a 3th-order 4-round integral property for Rijndael-160 as shown in Fig. 13. We have found 42 3th-order integral properties (essentially the shifted ones). We also have found the backward 3-th order backward integral property for Rijndael-160 shown in Fig. 14.

Using those two backward and forward properties, we could build a 7-round known key distinguisher shown in Fig. 15. The process is exactly the same than the one described in 3.1 and the time complexity is similar to the time it takes to do 2^{40} 7-round Rijndael-160 encryptions and the memory needed is small. If we also use the k-sum problem to estimate the corresponding time to find a k-sum

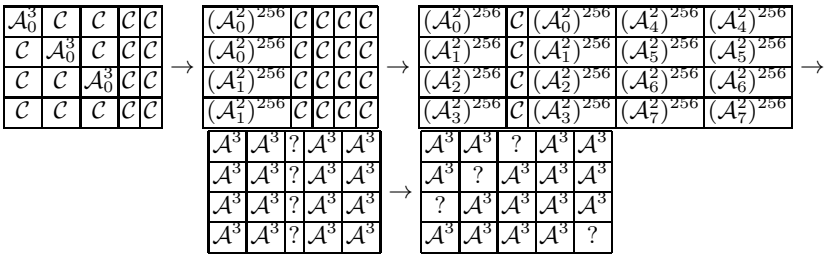


Fig. 13. 3th order 4-round forward integral property of Rijndael-160 (the last round is without MixColumns)

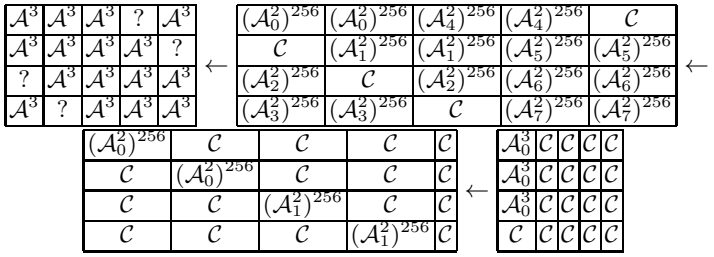


Fig. 14. 3th order integral backward property for Rijndael-160

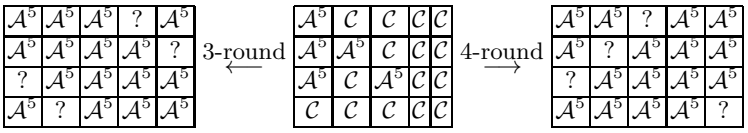


Fig. 15. The 7-round Rijndael-160 distinguisher with 2^{40} middle-texts. The 7th round is without MixColumns.

Table 2. Summary of known-key distinguishers on Rijndael-b. CM means Chosen Middle-texts.

Cipher	nb rounds	Key sizes	Data	Time Complexity	Memory	Source
AES	7	(all)	2^{56} CM	2^{56}	small	[12]
Rijndael-256	8	(all)	2^{40} CM	2^{40}	small	this paper
Rijndael-224	8	(all)	2^{72} CM	2^{72}	small	this paper
Rijndael-192	7	(all)	2^{32} CM	2^{32}	small	this paper
Rijndael-160	7	(all)	2^{40} CM	2^{40}	small	this paper

for a 160-bit permutation with $n = 160 - 32$ and $k = 2^{40}$, the corresponding complexity is around 2^{43} operations ignoring small constants and memory. Thus, we conjecture that we have found a known-key distinguisher for Rijndael-192 reduced to 7 rounds using 2^{40} middle-texts.

4 Conclusion

In this paper, we have shown how to build known-key distinguisher against the various versions of Rijndael-b using essentially particular new dth-order integral properties in forward and backward sense. Table 2 sums up the results presented in this paper.

Note that we have used as done in [12] the k-sum problem to estimate the corresponding complexity to build such distinguishers for random permutations. This model is less pertinent in our case because we need to estimate such a problem for random functions and no more for random permutations. Thus, we however think that the corresponding complexity is around to be the same even if this stays as an open problem.

References

1. Canetti, R., Goldreich, O., Halevi, S.: On the random oracle methodology, revisited. Journal of the ACM 51(4), 557–594 (2004)
2. Daemen, J., Knudsen, L.R., Rijmen, V.: The block cipher Square. In: Biham, E. (ed.) FSE 1997. LNCS, vol. 1267, pp. 149–165. Springer, Heidelberg (1997)
3. Daemen, J., Rijmen, V.: AES proposal: Rijndael. In: The First Advanced Encryption Standard Candidate Conference. N.I.S.T. (1998)
4. Daemen, J., Rijmen, V.: The Design of Rijndael. Springer, Heidelberg (2002)
5. Ferguson, N., Kelsey, J., Lucks, S., Schneier, B., Stay, M., Wagner, D., Whiting, D.: Improved cryptanalysis of rijndael. In: Schneier, B. (ed.) FSE 2000. LNCS, vol. 1978, pp. 213–230. Springer, Heidelberg (2001)
6. FIPS 197. Advanced Encryption Standard. Federal Information Processing Standards Publication 197, U.S. Department of Commerce/N.I.S.T (2001)
7. Galice, S., Minier, M.: Improving integral attacks against Rijndael-256 up to 9 rounds. In: Vaudenay, S. (ed.) AFRICACRYPT 2008. LNCS, vol. 5023, pp. 1–15. Springer, Heidelberg (2008)
8. Gilbert, H., Minier, M.: A collision attack on 7 rounds of Rijndael. In: AES Candidate Conference, pp. 230–241 (2000)
9. Nakahara Jr., J., de Freitas, D.S., Phan, R.C.-W.: New multiset attacks on Rijndael with large blocks. In: Dawson, E., Vaudenay, S. (eds.) Mycrypt 2005. LNCS, vol. 3715, pp. 277–295. Springer, Heidelberg (2005)
10. Junod, P.: On the optimality of linear, differential, and sequential distinguishers. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 17–32. Springer, Heidelberg (2003)
11. Knudsen, L.R.: Contemporary block ciphers. In: Damgård, I. (ed.) Lectures on Data Security. LNCS, vol. 1561, pp. 105–126. Springer, Heidelberg (1999)
12. Knudsen, L.R., Rijmen, V.: Known-key distinguishers for some block ciphers. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 315–324. Springer, Heidelberg (2007)

13. Knudsen, L.R., Wagner, D.: Integral cryptanalysis. In: Daemen, J., Rijmen, V. (eds.) FSE 2002. LNCS, vol. 2365, pp. 112–127. Springer, Heidelberg (2002)
14. Luby, M., Rackoff, C.: How to construct pseudorandom permutations from pseudorandom functions. SIAM Journal on Computing 17(2), 373–386 (1988)
15. Matyas, S.M., Meyer, C.H., Oseas, J.: Generating strong one-way functions with cryptographic algorithm. IBM Technical Disclosure Buletin 27, 5658–5659 (1985)
16. Preneel, B., Govaerts, R., Vandewalle, J.: Hash functions based on block ciphers: A synthetic approach. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 368–378. Springer, Heidelberg (1994)
17. Vaudenay, S.: Decorrelation: A theory for block cipher security. J. Cryptology 16(4), 249–286 (2003)

The GLUON family: a lightweight Hash function family based on FCSRs

Abstract. Since the beginning of the SHA3 competition, the cryptographic community has seen the emergence of a new kind of primitives: the lightweight cryptographic hash functions. At the time writing this article, two representatives of this category have been published: QUARK [7] and PHOTON [18] designed to match RFID constraints.
In this paper, we propose a third representative of this category which is called GLUON. It is based on the sponge construction model [11] as QUARK and PHOTON and inspired by two stream ciphers F-FCSR-v3 [4] and X-FCSR-v2 [10]. From the generic definition of our lightweight hash function, we derive three different instances according the required security level that must be reached.
For example, our lightest instance (GLUON-128/8) dedicated to 64-bit security level fits in 2071 gate-equivalents which stays competitive when compared with the parallel implementation of U-QUARK. The software performances are good for GLUON-224/32, our heaviest instance.
Keywords: lightweight hash function, FCSRs, sponge functions.

Introduction

The last five years have seen the emergence of new challenging tasks that consist in designing lightweight primitives dedicated to very constrained environments such as sensors or RFID tags. Among those proposals, many block ciphers such as PRESENT [13], HIGH [20], mCrypton [23] or KATAN & KTANTAN [15] have been specially designed to fit with a very compact hardware implementation.

The same kind of works concerning lightweight hash functions has just been initiated with two existing standalone proposals: QUARK [7] and PHOTON [18] both based on sponge constructions [11]. The need for such proposals comes first from the embedded system community (RFID and sensor) and second from the lack of lightweight SHA3 finalists [26]. Indeed, all the SHA3 finalists requires more than 12000 GE for a 128-bit security level. Some previous proposals such as SQUASH [29] or ARMADILLO [8] have been done but show their respective weaknesses [28, 1]. The first step in the design of lightweight hash function dates from the use of the block cipher PRESENT in the Davies-Meyer mode of operation. However, and as already noticed in [7, 18], sponge constructions allow a better ratio between internal state size and security level, better than for traditional modes of operations even if hashing small messages is not really efficient due to the squeezing step.

Following the recent proposals for lightweight hash functions taking their name from small particles[1], we propose a new lightweight hash function family

[1] see: http://www.131002.net/data/talks/quarks_rump.pdf

called GLUON based on a sponge construction. This new family is based on a particular Feedback with Carry Shift Register (FCSR), elementary building block well studied during the two last decades. Even if the hardware size of such a primitive is a little bit heavier than the basic building blocks used in QUARK and PHOTON, we think that the well-known design principles of FCSRs could be considered as a strength of our proposed design.

This paper is organized as follows: in Section 1, we recall the definitions of sponge constructions and of word ring FCSRs. In Section 2, we describe the underlying function that composes the sponge construction based on a word ring FCSR. In Section 3, we give some insights about our chosen design whereas in Section 4 we sum up all the security observations we made concerning the f function. In Section 5, we provide performance results for hardware and software implementations. Finally, Section 6 concludes this paper.

1 Background

1.1 Sponge constructions

Sponge constructions have been proposed by Bertoni et al. in [11] as a new way of building hash functions from a fixed function or a fixed permutation. A sponge construction as defined in Fig. 1 has a *rate* r which corresponds with the block length, a *capacity* cp and an *output length* N. The *width* of its internal state b is defined as $r + cp$ that must be greater than N.

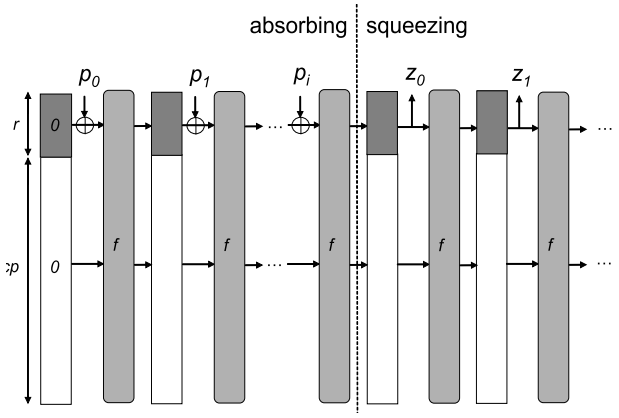


Fig. 1. The sponge construction.

Given an initial state, the sponge construction processes a message M of length $|M| = \log_{2^r}(M)$ as follows:

1. **Initialization:** The message is padded by appending a '1' bit and sufficiently many zeros to reach a length multiple of r.
2. **Absorbing phase:** The r-bit message blocks are xored into r bits of the state interleaved with applications of the function f.
3. **Squeezing phase:** Some r bits of the state are returned as output, interleaved with applications of the function f, until N bits are returned.

1.2 FCSR automata in word ring representation

Usually speaking, a FCSR as defined in [17, 22] consists of a binary main register and of a carry register but contrary to LFSRs the performed operations are no more xors over $\mathbb{F}_2$ but additions with carry in the set of 2-adic integers $\mathbb{Z}_2$ (i.e. the set of power series: $\sum_{i=0}^{\infty} s_i 2^i$, $s_i \in \{0,1\}$). Each cell of the main register produces a sequence $S = (s_n)_{n\in\mathbb{N}}$ that is eventually periodic if and only if there exist two numbers p and q in $\mathbb{Z}$, q odd, such that $s = p/q$. This sequence is strictly periodic if and only if $pq \leq 0$ and $|p| \leq |q|$. The period of S is the order of 2 modulo q, i.e., the smallest integer P such that $2^P \equiv 1 \pmod{q}$. The period satisfies $P \leq |q| - 1$. If q is prime and if $P = |q| - 1$, the sequence S is called an ℓ-sequence. ℓ-sequences have many proved properties that could be compared to the ones of m-sequences: known period, good statistical properties, fast generation, etc.

F. Arnault et al. have studied in [4] and in [10] efficient hardware and software FCSRs using matrix representations. They show the following results:

Definition 1. *A (diversified or ring) FCSR is an automaton composed of a main shift register of n binary cells $m = (m_0, \ldots, m_{n-1})$, and a carry register of n integer cells $c = (c_0, \ldots, c_{n-1})$. It is updated using the following relations:*

$$\begin{cases} m(t+1) = Tm(t) + c(t) & \bmod 2 \\ c(t+1) = Tm(t) + c(t) & \div 2 \end{cases} \tag{1}$$

where T is a $n \times n$ matrix with coefficients 0 or 1 in $\mathbb{Z}$, called transition matrix.

Note that $\div 2$ is the traditional expression: $X \div 2 = \frac{X - (X \bmod 2)}{2}$.

They also prove the following property:

Theorem 1 ([4] Theorem 1). *The series $\boldsymbol{M}_i(t)$ observed in the cells of the main register are 2-adic expansion of p_i/q with $p_i \in \mathbb{Z}$ and with $q = \det(I - 2T)$.*

The T transition matrix completely defines the ring FCSR as shown in Theorem 1. Moreover and as shown in [4], ring FCSRs have better hardware implementations than classical Galois or Fibonacci FCSRs. They also improve the diffusion of differences because these diffusions could be computed as the diameter d of the graph associated to the transition matrix T. Typically this value is close to $n/4$ instead of n in the Galois or Fibonacci cases.

In [10], Berger et al. describe word ring FCSRs that are efficient both in hardware and in software. Those FCSRs are completely determined by the choice of the matrix T. Contrary to ring FCSRs, word ring FCSRs act on words of size

r bits depending on the targeted architecture. Classically, r could be equal to 8, 16, 32 or 64 bits. Then, in the generic description of a word ring FCSR, the associated matrix T is defined on r-bit words. The main register of this kind of FCSR could thus be represented as w r-bit words $m_0, \cdots, m_{w-1}$ with feedback words $c_0, \cdots, c_{w-1}$. The deduced T matrix is of size $w \times w$. For example, the following T matrix represents a word ring FCSR acting on $r = 8$ bits words:

$$T = \begin{pmatrix} 0 & I & 0 & SL^3 & 0 \\ 0 & 0 & I & 0 & SR^2 \\ 0 & 0 & SL^1 & I & 0 \\ SR^3 & 0 & 0 & 0 & I \\ I & 0 & 0 & 0 & 0 \end{pmatrix}$$

where I is the $r \times r$ identity matrix, the SL^a operation is the left shift at 8-bit level by a bits, SR^b is the right shift operation at 8-bit level by b bits (two other operations could also be used: RL^d, the rotation on the left by d bits and RR^e, the rotation on the right by e bits).

This matrix defines the associated word ring FCSR described in Figure 2 with $n = 40$ and $r = 8$.

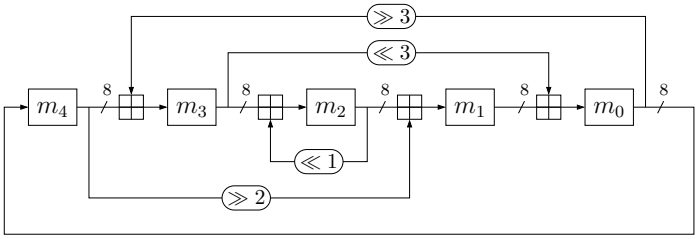


Fig. 2. A FCSR with efficient software design

The corresponding q value could be directly computed using the formula given in Theorem 1 and is equal to -1497813390989. This number is prime and has a maximal order. Thus the corresponding FCSR produces ℓ-sequences, and is not only efficient in software due to the word oriented structure, it is also efficient in hardware (here only 24 binary additions are required) because the intrinsic word ring nature of this FCSR representation limits the number of required gates.

2 Description of the GLUON Hash family

The GLUON family is based on a sponge construction where the f function calls a filtered FCSR. The filtered FCSR is directly inspired from the F-FCSR-v3 hardware stream cipher [4] and by the X-FCSR-v2 software stream cipher

[10]. All the proposed instances of the GLUON family varies according the version but all versions are based on "ring-word FCSRs" to be efficient not only in hardware but also in software. The general structure of the GLUON family is described on Fig. 3. The elementary building blocks are the following ones:

- The content of the word ring FCSR of size w is denoted $m(t) = (m_0(t), \cdots, m_{w-1}(t))$ for the main register where w is the length in words of the considered FCSR and $c(t) = (c_0(t), \cdots, c_{w-1}(t))$ for the carry register with a active memories (i.e. a internal feedbacks). The FCSR is defined by its associated matrix T as seen in the previous section.
- A filter FI is also defined to filter the content of the main register $m(t)$. It is xor-linear to break the 2-adic structure of the automaton. As done in [4], it consists in xoring together some shifted version of the words of the main register that have active carries. More precisely, let $\mathcal{F} = \{m_{f_0}, \cdots, m_{f_{\ell-1}}\}$ be the set of all the words m_i that have a feedback. Then, r bits of output are: $\forall\ 0 \leq i < r,\ FI = \bigoplus_{j \equiv i \mod r} (m_{f_j})_{<<<k}$ with k a value in the set $[-r/2, +r/2]$. This linear filter is only used at the end of the computation of the function to extract the output from the state of the FCSR.

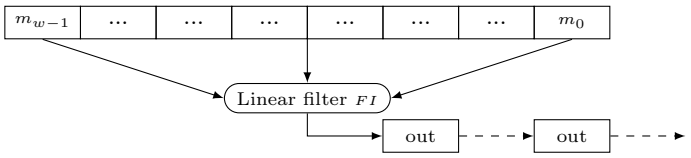


Fig. 3. General view of the f function of the GLUON family.

The size of the input/output of f is $b = (w-1) \times r$ (instead of $w \times r$ as expected because a particular word (the word $w-1$) of the main register is not used as input of the function).

2.1 Details of the f function

f processes a b-bit input in three steps:

Initialization: For an input $s = (s_0, \cdots, s_{b-1})$, f initializes its internal state as follows:
- The $w-1$ first words of m are initialized with the $(w-1) \times r$ input bits.
- The last word of m is initialized with the all-one string of length r.
- The carry register c is initialized with the all-zero string.

State Update: From the internal state $(m(t), c(t))$, the FCSR is clocked $d+4$ times using its internal transition function:

$$m(t+1) = Tm(t) + c(t) \mod 2$$
$$c(t+1) = Tm(t) + c(t) \div 2$$

Computation of the output: The FCSR is then clocked $w-1$ times. At each iteration, a r-bit word is extracted with the linear filter F in order to obtain the $(w-1) \times r$ bits of output.

2.2 The sponge construction deduced from the *f* function

The rate of the associated sponge construction is r, its capacity is $cp = (w-2) \times r$. The size of f is $b = r + cp$. Let us now analyze in few words how this f function fits in the sponge model. The external part of the f function is composed of the first r-bit word of the input whereas the internal part consists of the following $(w-2)$ r-bit words of the input. In other words, during the absorbing steps, at each call of f, a message word p_i of length r bits is xored with the first r-bit word of the FCSR main register. Then, this first r-bit word is output when the squeezing step begins.

2.3 Proposed instances

As done for QUARK and PHOTON, we propose 3 different instances of our GLUON family functions that we will denote GLUON-N/r (the same that for QUARK).

- *64-bit security level:* $r = 8$, $c = 128$, $b = 136$, $N = 128$ leading to an FCSR composed of 18 8-bit words and with about 70 carries. The complete description of the Matrix T and of the filter FI is given in Appendix A.
- *80-bit security level:* $r = 16$, $c = 160$, $b = 176$, $N = 160$ leading to an FCSR composed of 12 16-bit words and with about 90 carries. The complete description of the Matrix T and of the filter FI is given in Appendix B.
- *112-bit security level:* $r = 32$, $c = 224$, $b = 256$, $N = 224$ leading to an FCSR composed of 9 32-bit words and with about 130 carries. The complete description of the Matrix T and of the filter FI is given in Appendix C.

As done for PHOTON, we could also add some other instances with as possible output sizes 80 and 256. In those cases, the corresponding r values are respectively 8 and 32.

3 Design Rationale

3.1 Flat Sponge Claim

Sponge construction is a recent model for iterated hash functions and random number generation developed in [11] and in [12]. The general security claim is done in the indifferentiability framework introduced by Maurer, Renner and Holenstein [24]. In [11], the authors prove that the success probability of differentiating a sponge construction calling a random permutation or transformation from a random oracle is upper bounded by $1 - \exp(-Q^2 2^{-(c+1)})$ with Q the number of calls to f or its inverse when possible.

From this particular bound, the authors of [11] deduce a simplification of the proof model which considers from the previous result only the worst-case success probability. This simplification called the *flat sponge claim* makes the security of a concrete function seen as a random sponge depend on the capacity of the random sponge. More precisely, the collision resistance of a sponge construction under the flat sponge claim is $2^{\min(cp,n)/2}$, the (second) preimage resistance is $2^{\min(cp/2,n)}$.

Thus, we design our function using this flat sponge argument which allows to minimize the internal state (looking at the indifferentiability in the random oracle model) of the used function when compared with other traditional hash constructions such as the Merkle-Damgård one for example. In fact, sponge constructions lead to achieve the highest security level that could be obtained for a hash construction.

3.2 Choice of the f function

As already mentioned for the hash function Quark [7], the design choice of f comes from this simple idea: from a stream cipher with an internal state of size n, one can construct a function from $\{0,1\}^b$ into itself as follows:

- The b-bit input is padded to an initial state of size n bits,
- The stream cipher is initialized as usual,
- The first b output bits compose the output of the f function.

Under the hypothesis that the stream-cipher is "perfect", the f function looks like a random function. In other words, finding a bias in the function f is equivalent to find a weakness of the stream-cipher.

F-FCSR-v3 and X-FCSR-v2. We have based our f design on a mix between two FCSR based stream ciphers which are F-FCSR-v3 [4] and X-FCSR-v2 [10]. The first one is dedicated to hardware and uses a linear filter as done here whereas the second one introduces the word ring oriented structure of an FCSR which is efficient both in hardware and in software. This kind of automata is the building block of our f design.

Since the first proposal of a stream cipher based on an FCSR in 2005 in [2], the security of such designs has been carefully studied through the eStream call for stream ciphers [27]. One of the first proposals called F-FCSR-v2 based on a Galois FCSR was however successfully attacked by M. Hell and T. Johansson in 2008 in [19]. This attack exploits a particular dependence between the feedback bit and all the carry bits. This attack is also efficient against the first version of X-FCSR (X-FCSR-v1) as shown in [30]. Those particular attacks lead to the modification of the original proposals into the new versions F-FCSR-v3 and X-FCSR-v2 based on new matrix representations of the FCSR leading to discard the two previous attacks as detailed in Section 4. Since their publications, two years ago, no attack has been exhibited against the two new stream ciphers which are based on strong security arguments. This leads to have a relative level

of confidence concerning those two particular primitives, particularly if we look at the absence of existing distinguishers against the building blocks.

Moreover, those designs are simple and relatively efficient both in software and hardware. Thus, these simple arguments concerning the efficiency of F-FCSR-v3 and of X-FCSR-v2 and the security level claimed lead to naturally consider them as possible instance for a compression function in the sponge model.

Quality of the f function. The good statistical properties (period, balanced sequences and so on) of the underlying building blocks come from the 2-adic properties and are equivalent over $\mathbb{Z}_2$ to the ones of LFSRs over $\mathbb{F}_2$. We will see now how the design choices made here are efficient to prevent classical attacks.

First, the number of clocks in f is $d+4$ to ensure a complete diffusion of the message block into all the words of the FCSR. Even if a difference is introduced in the message block, this difference will influence all the output blocks. This is due to the diameter definition which clearly improves the diffusion speed in a word ring representation. In other words, the diffusion is complete after $d+4$ clocks.

Let us consider an FCSR with connection integer q which produces ℓ-sequences, i.e. such that the order of 2 modulo q is $P = |q|-1$. After a few iterations from an initial state, the automaton is in a periodic sequence of states of length P. The average number of required iterations to be in such a state is experimentally less than $log_2(n)$, where n is the size of the main register (see [5] for more details). Thus, during the application of f such states are always reach because for all our cases $log_2(n) < d$. Those particular periodic states are all on a (the) main cycle of size P, generally this value is close to n (equal to $(w+1)r$ in our design). This leads to consider a function f which is really close to a permutation from $\{0,1\}^b$ into itself because the surjective part of the construction is really limited once the function f acts on the main cycle. In other words, we exploit here the fact that the transition function of an FCSR becomes a permutation on its periodic states.

So, our f function could be considered as an application from $\{0,1\}^b$ into the set of periodic states, i.e. into a set of size $\{0,1\}^n$ that has a behavior close to a permutation leading to a very low collision probability. Indeed, each possible input value conducts to a particular part of the main cycle. The entering function is thus a quasi-permutation on its cycles with a very small loose of entropy. As soon as the main cycle is reached, f does not anymore loose entropy even after several iterations. The entropy lost that could be considered here comes from the extraction step. Thus, in summary, the design of f prevents any entropy loss as for a true random function.

However, the generic word ring FCSR structure brings with it a particular problem: there is on the graph representation of the FCSR two particular fixed points which are the all-zero point and the all-one point. Thus, to prevent attacks producing collisions on two different messages M and M' which differ on a certain number of all-zero blocks at the beginning (i.e. $M = 0||0||m_0$ and $M' = 0||m_0$

for example which lead to the same output results), we introduce the all-one bit constant in a word of the main register to avoid the all-zero point. The all-one point is discarded by the initialization of the carry register to the all-zero word at each f application. This state can never be reached by design.

4 Security Analysis

As already explained in Section 3.1, the resistance of the construction against classical hash function attacks ((second) preimage attack, collision attack) is proved in the sponge model. So the bounds for the different attacks are the ones given in Section 3.1. We discussed here the resistance of the f function against traditional attacks that target the underlying function f itself.

Cube attacks and Cube testers. Recently, new kind of attacks called "Cube attacks" have appeared (the most recent reference in this area is [6]). Those attacks could be efficient as soon as "Cube testers" show their efficiency in simplifying a part of the ANF of a function. Thus, those attacks could be applied against functions which have particular weaknesses in their algebraic structure. As shown in [9] for the case of a Galois FCSR, such a particular structure (with a low degree component) does not exist in a Galois FCSR. This is the same thing for a word ring FCSR and thus for the f function chosen here. Indeed, for example with a Galois FCSR of size 16 after 7 clocks, the number of monomials in the output algebraic equations is 125420 with a degree of at least 10 considering only the variables of the main register as unknowns. The number of monomials of the obtained system becomes huge with a high degree as mentioned in [9]. It discards the potential use of cube testers and renders cube attacks harmless.

The Hell and Johansson attack: LFSRization of the FCSR. As noticed in [4], recent attacks against FCSRs and F-FCSRs described in [19] and [16] are discarded by the use of ring FCSRs as done in [4] and the use of word ring FCSRs in [10]. More precisely, the attack of [19, 30] against F-FCSR using Galois representation exploits the control of the feedback bit over all the other feedbacks. This relation is no more true in the case of a ring FCSR: the linear behavior is observed with probability about 2^{-t} during t clocks for a Galois FCSR whereas it becomes $2^{-t \cdot k}$ for k feedbacks for a ring or a word ring FCSR. The attack presented in [16] is also discarded because it only concerns stream ciphers where the underlying building block is a Fibonacci FCSR which is not the case here.

Concerning other potential linear attacks, as in the case of stream ciphers, the correlation and fast correlation attacks have been proven difficult to mount in [3, 4] due to the inherent non-linearity of all kind of possible FCSRs, we conjecture that due to the design of the f function the same arguments could be used to discard the existence of linear relations for this function.

Differential attacks. Differential attacks stay a very powerful tool to find inner collisions in compression functions as shown with MD5 and SHA-1 [31]. The main idea relies on the non-ideal behavior of difference propagations through the compression function.

The resistance of F-FCSR stream ciphers against differential attacks have been proven after some changes in F-FCSR-H v2 due to a slow diffusion of differences presented in [21]. More precisely, in [21], the authors show that a difference introduced in some cell of the FCSR automaton remains localized as long as this difference does not reach the feedback end of the register. From this remark, they deduce an attack on the IV process of F-FCSR-H v1. Thus, in [3], The designers prevent this attack from happening by redesigning the IV setup and increasing the number of initial clocks to be sure that a sufficient diffusion of differences occurs.

In the new ring and word ring design, the diffusion criterion allows to determine a minimal value (that corresponds with the diameter of the graph d) from which a sufficient diffusion level occurs. This is why, the number of clocks of the f function is $d + 4$: we are sure that a minimal diffusion (in general and for differences in particular) is reached.

Slide distinguishers. As mentioned in Section 3.2, slide distinguishers could be built on the two particular fix points of the graph of an FCSR which are the all-zero point transformed into itself and the all-one point also transformed into itself. The all-one point could not be reached for f when used inside the sponge construction due to the way the sponge construction builds its internal states. However, the all-zero point could be reached for f if a particular initial value is not given to a particular word of the main register of the FCSR. This is why, to discard slide distinguishers that make use of block messages equal to zero, a particular word of the main register is initialized to the all-one point.

In conclusion, we could say that since the beginning of FCSR study in the cryptographic context 6 years ago, all the original awkwardnesses leading to conventional attacks or to more tricky attacks have been discarded along the design process. This is why, we think that cryptanalyzing ring FCSRs passes through creating new attacks exploiting really original sorts of relations that are no more linear, differential or algebraic.

5 Performances

This section details our experimental results. We describe the tools used and compare our results to previous works.

5.1 Hardware performances

This section reports our hardware implementation of the GLUON instances. For simulation, we used the *ModelSim PE* simulator [25], version *10,0a*. For

synthesis, we used the ASIC synthetisor *Cadence RTL* version *RC10.1.101* (32-bits version) [14].

We implemented the three variants detailed in Section 2.3. In our design, we load the FCSR main register M in parallel, not serially, which means that we use a little more gates, but gain in efficiency. Moreover, the output of the filter is directly loaded into M in order to avoid storing information in the filter and copy it later. Another optimization we made was, in the FCSR, to avoid storing bits that always worth '0' because values are shifted.

Results are reported in Table 1. As we can see, our proposals compete well in terms of area requirements compared to the QUARK proposals since they are 13.4% smaller for the 64-bits version and similar for the 80-bits and 112-bits versions. We did not include power consumption because the power consumption strongly depends on the technology used and cannot be compared between different technologies in a fair manner. In addition, simulated power results strongly depend on the simulation method used, and the effort spent.

Hash function	Security		Block	Area	Lat.	Thr.
	Pre.	Coll.	[bits]	[GE]	[cycles]	kbps
GLUON-64	128	64	8	2071	62	
GLUON-80	160	80	16	2799.3	46	
GLUON-112	224	112	32	4724	51	
U-QUARK×8	128	64	8	2392	68	11.76
D-QUARK×8	160	80	16	2819	88	18.18
S-QUARK×16	224	112	32	4640	64	50.00

Table 1. Compared hardware performances of GLUON with previous works

5.2 Software performances

This section reports our software implementations. Table 2 gives the software performances we obtained on the variants detailed in Section 2.3. The processor used for the benchmarks is an Intel Core 2 Duo clocked at 2.66 GHz. Note that our implementation is not optimzed for a particular processor, *e.g.* it is not optimized for running on a 64-bits processor. Results are good but difficult to compare to the 8k, 30k and 22k cycles per byte of U-QUARK, D-QUARK and S-QUARK benchmarked in [18].

GLUON-64	GLUON-80	GLUON-112
7162	1970	446

Table 2. Software performances in cycles per byte of the GLUON variants for long messages

6 Conclusion

After the lightweight hash proposals QUARK and PHOTON, the family of lightweight particles expands with the GLUON family and three particular instances . GLUON-128/8, GLUON-160/16 and GLUON-224/32. Even if the software and hardware performances of GLUON are worst than the ones of PHOTON, there are comparable when targeting hardware to the parallelized versions of QUARK. We think that our design is relevant and of real interest because the basic building blocks have been well-studied since twenty years.

We do not develop here (as done for PHOTON) the case where the squeezing step is reduced and produces more output blocks at each step. An initial study concerning this aspect shows that we could transform GLUON-128/8 into a more challenging version fitting with the requirements of such a modified version. The idea is to directly output, during the squeezing step, one 8-bit word at each clock without waiting for $d + 4$ clocks and without changing the input of the f function at each f call. Of course, due to its simplicity, this version is more risky but the reached security level could be compared to the one of F-FCSR-v3.

References

1. Mohamed Ahmed Abdelraheem, Céline Blondeau, María Naya-Plasencia, Marion Videau, and Erik Zenner. Cryptanalysis of armadillo2. Cryptology ePrint Archive, Report 2011/160, 2011. http://eprint.iacr.org/.
2. François Arnault and Thierry P. Berger. F-FCSR: design of a new class of stream ciphers. In *Fast Software Encryption - FSE 2005*, Lecture Notes in Computer Science 3557, pages 83–97. Springer-Verlag, 2005.
3. François Arnault, Thierry P. Berger, and Cédric Lauradoux. Update on F-FCSR Stream Cipher. ECRYPT - Network of Excellence in Cryptology, Call for stream Cipher Primitives - Phase 2 2006. http://www.ecrypt.eu.org/stream/.
4. François Arnault, Thierry P. Berger, Cédric Lauradoux, Marine Minier, and Benjamin Pousse. A new approach for fcsrs. In *Selected Areas in Cryptography - SAC 2009*, volume 5867 of *Lecture Notes in Computer Science*, pages 433–448. Springer, 2009.
5. François Arnault, Thierry P. Berger, and Marine Minier. Some Results on FCSR Automata With Applications to the Security of FCSR-Based Pseudorandom Generators. *IEEE Transactions on Information Theory*, 54(2):836–840, 2008.
6. Jean-Philippe Aumasson, Itai Dinur, Willi Meier, and Adi Shamir. Cube testers and key recovery attacks on reduced-round md6 and trivium. In *Fast Software Encryption - FSE 2009*, volume 5665 of *Lecture Notes in Computer Science*, pages 1–22. Springer, 2009.
7. Jean-Philippe Aumasson, Luca Henzen, Willi Meier, and María Naya-Plasencia. Quark: A lightweight hash. In *Cryptographic Hardware and Embedded Systems, CHES 2010*, volume 6225 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2010.
8. Stéphane Badel, Nilay Dagtekin, Jorge Nakahara, Khaled Ouafi, Nicolas Reffé, Pouyan Sepehrdad, Petr Susil, and Serge Vaudenay. Armadillo: A multi-purpose cryptographic primitive dedicated to hardware. In *Cryptographic Hardware and Embedded Systems, CHES 2010*, volume 6225 of *Lecture Notes in Computer Science*, pages 398–412. Springer, 2010.

9. Thierry P. Berger and Marine Minier. Two algebraic attacks against the f-fcsrs using the iv mode. In *Progress in Cryptology - INDOCRYPT 2005*, volume 3797 of *Lecture Notes in Computer Science*, pages 143–154. Springer, 2005.
10. Thierry P. Berger, Marine Minier, and Benjamin Pousse. Software oriented stream ciphers based upon fcsrs in diversified mode. In *Progress in Cryptology - INDOCRYPT 2009*, volume 5922 of *Lecture Notes in Computer Science*, pages 119–135. Springer, 2009.
11. Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. On the indifferentiability of the sponge construction. In *Advances in Cryptology - EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 181–197, 2008.
12. Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. Sponge-based pseudo-random number generators. In *Cryptographic Hardware and Embedded Systems, CHES 2010*, volume 6225 of *Lecture Notes in Computer Science*, pages 33–47. Springer, 2010.
13. Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and C. Vikkelsoe. PRESENT: An Ultra-Lightweight Block Cipher. In *Cryptographic Hardware and Embedded Systems - CHES 2007*, Lecture Notes in Computer Science 4727, pages 450–466. Springer Verlag, 2007.
14. Cadence. Encounter rtl compiler. http://www.cadence.com/products/ld/rtl_compiler.
15. Christophe De Cannière, Orr Dunkelman, and Miroslav Knezevic. KATAN and KTANTAN - A Family of Small and Efficient Hardware-Oriented Block Ciphers. In *Cryptographic Hardware and Embedded Systems - CHES 2009*, Lecture Notes in Computer Science 5747, pages 272–288. Springer Verlag, 2009.
16. Simon Fischer, Willi Meier, and Dirk Stegemann. Equivalent Representations of the F-FCSR Keystream Generator. In *ECRYPT Network of Excellence - SASC Workshop*, pages 87–94, 2008. Available at `http://www.ecrypt.eu.org/stvl/sasc2008/`.
17. Mark Goresky and Andrew Klapper. Periodicity and distribution properties of combined fcsr sequences. In *Sequences and Their Applications - SETA 2006*, volume 4086 of *Lecture Notes in Computer Science*, pages 334–341. Springer, 2006.
18. Jian Guo, Thomas Peyrin, and Axel Poschmann. The photon family of lightweight hash functions. In *CRYTPO 2011*, volume to appear of *Lecture Notes in Computer Science*. Springer, 2011.
19. Martin Hell and Thomas Johansson. Breaking the F-FCSR-H stream cipher in real time. In *Advances in Cryptology - ASIACRYPT 2008*, volume 5350 of *Lecture Notes in Computer Science*, pages 557–569. Springer, 2008.
20. Deukjo Hong, Jaechul Sung, Seokhie Hong, Jongin Lim, Sangjin Lee, Bonseok Koo, Changhoon Lee, Donghoon Chang, Jaesang Lee, Kitae Jeong, Hyun Kim, Jongsung Kim, and Seongtaek Chee. HIGHT: A New Block Cipher Suitable for Low-Resource Device. In *Cryptographic Hardware and Embedded Systems - CHES 2006*, Lecture Notes in Computer Science 4249, pages 46–59. Springer Verlag, 2006.
21. Éliane Jaulmes and Frédéric Muller. Cryptanalysis of the f-fcsr stream cipher family. In *Selected Areas in Cryptography - SAC 2005*, volume 3897 of *Lecture Notes in Computer Science*, pages 20–35. Springer, 2005.
22. Andrew Klapper and Mark Goresky. 2-adic shift registers. In *Fast Software Encryption - FSE'93*, Lecture Notes in Computer Science 809, pages 174–178. Springer-Verlag, 1993.

23. Chae Hoon Lim and Tymur Korkishko. mCrypton - A Lightweight Block Cipher for Security of Low-Cost RFID Tags and Sensors. In *Workshop on Information Security Applications - WISA 2005*, Lecture Notes in Computer Science 3786, pages 243–258. Springer Verlqg, 2005.
24. Ueli M. Maurer, Renato Renner, and Clemens Holenstein. Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In *Theory of Cryptography, First Theory of Cryptography Conference, TCC 2004*, volume 2951 of *Lecture Notes in Computer Science*, pages 21–39. Springer, 2004.
25. ModelSim. Modelsim pe - simulation and debug. `http://model.com/content/modelsim-pe-simulation-and-debug`.
26. National Institute of Standards and Technology. Announcing Request for Candidate Algorithm Nominations for a NewCryptographic Hash Algorithm (SHA-3) Family. *Federal Register*, 27(212):62212–62220, November 2007. Available: `http://csrc.nist.gov/groups/ST/hash/documents/FR_Notice_Nov07.pdf`(2008/10/17).
27. Network of Excellence in Cryptology ECRYPT. Call for stream cipher primitives. `http://www.ecrypt.eu.org/stream/`.
28. Khaled Ouafi and Serge Vaudenay. Smashing squash-0. In *Advances in Cryptology - EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 300–312. Springer, 2009.
29. Adi Shamir. Squash - a new mac with provable security properties for highly constrained devices such as rfid tags. In *Fast Software Encryption - FSE 2008*, volume 5086 of *Lecture Notes in Computer Science*, pages 144–157. Springer, 2008.
30. Paul Stankovski, Martin Hell, and Thomas Johansson. An efficient state recovery attack on X-FCSR-256. In *Fast Software Encryption - FSE 2009*, volume 5665 of *Lecture Notes in Computer Science*, pages 23–37. Springer, 2009.
31. Xiaoyun Wang and Hongbo Yu. How to break md5 and other hash functions. In *Advances in Cryptology - EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 19–35. Springer, 2005.

A Matrix T and FI function for the 64-bit security level GLUON version

A.1 Matrix T

Parameters: $w = 18$ blocks of $r = 8$ bits. The value of q is equal to:

$$-22902874254594039368339911884338634654967949$$

```
> SL;
[0 1 0 0 0 0 0 0]
[0 0 1 0 0 0 0 0]
[0 0 0 1 0 0 0 0]
[0 0 0 0 1 0 0 0]
[0 0 0 0 0 1 0 0]
[0 0 0 0 0 0 1 0]
[0 0 0 0 0 0 0 1]
[0 0 0 0 0 0 0 0]
```

```
> SR;
[0 0 0 0 0 0 0 0]
[1 0 0 0 0 0 0 0]
[0 1 0 0 0 0 0 0]
[0 0 1 0 0 0 0 0]
[0 0 0 1 0 0 0 0]
[0 0 0 0 1 0 0 0]
[0 0 0 0 0 1 0 0]
[0 0 0 0 0 0 1 0]
```

$$T = \begin{pmatrix}
0 & I & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & SR^3 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & I & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & I & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & I & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & SL^6 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & I & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & I & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & I & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & I & 0 & 0 & SR^3 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & I & 0 & 0 & 0 & 0 & SR^3 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & I & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & SL & I & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & I & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & SL^6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & I & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & I & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & I & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & SL^3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & I & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & I \\
I & SR^6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}$$

$$m(t) = (m_0(t), \cdots, m_{17}(t))^T \text{ and } m(t+1) = Tm(t)$$

8-bit words with carries: m_0, m_3, m_7, m_8, m_{10}, m_{12}, m_{15}, m_{17}. The diameter d is equal to 44.

16

A.2 Filter FI

$$\begin{aligned} FI(m(t)) = & (m_0(t) \oplus (m_3(t)_{>>>4})) & \oplus \\ & (m_7(t) \oplus (m_8(t)_{<<<2}))_{>>>5} & \oplus \\ & (m_{10}(t) \oplus (m_{12}(t)_{>>>3}))_{<<<1} & \oplus \\ & (m_{15}(t) \oplus (m_{17}(t)_{>>>5}))_{<<<4} \end{aligned}$$

B Matrix T and FI function for the 80-bit security level GLUON version

B.1 Matrix T

Parameters: $w = 12$ blocks of $r = 16$ bits. The value of q is equal to:

$$-1196184691751347078913349747445350417991621293568616511077 3$$

$$T = \begin{pmatrix} 0 & I & 0 & 0 & 0 & 0\,0\,0 & 0 & 0 & SL^{12} & 0 \\ 0 & 0 & I & 0 & 0 & 0\,0\,0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & I & SR & 0\,0\,0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & I & 0\,0\,0 & SR^6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & I\,0\,0 & 0 & 0 & 0 & 0 \\ 0 & I & 0 & 0 & 0 & 0\,I\,0 & 0 & 0 & 0 & 0 \\ SR^6 & 0 & 0 & 0 & 0 & 0\,0\,I & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0\,0\,0 & I & 0 & 0 & 0 \\ 0 & 0 & 0 & SL^{10} & 0 & 0\,0\,0 & 0 & I & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0\,0\,0 & 0 & 0 & I & 0 \\ 0 & 0 & 0 & 0 & 0 & 0\,0\,0 & 0 & 0 & 0 & I \\ I & 0 & SL^{11} & 0 & 0 & 0\,0\,0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

16-bit words with carries: m_0, m_2, m_3, m_6, m_8, m_{11}. The diameter d is equal to 34.

B.2 Filter FI

$$\begin{aligned} F(m) = & (m_0(t) \oplus (m_2(t)_{>>>3})) & \oplus \\ & (m_3(t) \oplus (m_6(t)_{<<<5}))_{>>>7} & \oplus \\ & (m_8(t) \oplus (m_{11}(t)_{>>>2}))_{>>>2} \end{aligned}$$

C Matrix T and FI function for the 112-bit security level GLUON version

C.1 Matrix T

Parameters: $w = 9$ blocks of $r = 32$ bits. The value of q is equal to:

$$-42288419239352779381136581495386884385959973$$
$$562238326720894124651392664773627289720436 13$$

$$T = \begin{pmatrix} SR^{12} & I & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & I & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & I & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & I & 0 & SR^{9} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & I & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & I & SR^{7} & 0 \\ 0 & 0 & 0 & SL^{7} & 0 & 0 & 0 & I & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & I \\ I & 0 & SL^{6} & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

32-bit words with carries: m_0, m_3, m_5, m_6, m_8. The diameter d is equal to 42.

C.2 Filter FI

$$F(m) = (m_0(t) \oplus (m_3(t)_{>>>15})) \quad \oplus \\ (m_5(t) \oplus (m_6(t)_{<<<8}))_{>>>3} \oplus m_8(t)_{>>>13}$$

Annexe B

Articles présentés dans la deuxième partie

A trust protocol for community collaboration

Samuel Galice[1], Marine Minier[1], and Stéphane Ubéda[1]

CITI INSA-Lyon - ARES INRIA Project
CITI, INSA de Lyon, Bâtiment Léonard de Vinci
21 Avenue Jean Capelle, 69621 Villeurbanne Cedex
FirstName.Name@insa-lyon.fr

Abstract. In ambient environments, new security challenges that are not adequately addressed by existing security models appear. In such context, intelligent communication devices participate to spontaneous and self-organized networks where unexpected interactions with unknown devices took place. Without centralized organization, security turns in a risk management problem.
In this paper we propose and analyze a computational model of trust that captures trust dynamics of the human society. In our model, past experiences and recommendations are aggregated in the notion of *history of past interactions* which are protected by cryptographic material. To avoid the trust dissemination, each entity is viewed as an autonomous device and a trust level is computed based only upon selfish evaluation of common trustworthy nodes. Our proposal reduces the complexity of the decision-making process by providing *proved data* that can be the foundation of the final decision. The proposed trust model is described together with an overview of the cryptographic protocol and its security analysis. The trust function is analyzed through intensive simulations depending on the impact of the chosen parameters of the trust evaluation and on the dynamics of the studied groups.
Keywords: trust management framework, cryptographic protocol, Identity-based cryptosystems.

1 Introduction

Nowadays, the smart devices such as mobile phones, personal digital assistants and the like, act in a more and more ubiquitous environment. Their wireless communications capabilities grow up very quickly like their computing capacities. New types of services come out from dynamic groups of objects which can act together cooperatively facing various interaction contexts.

Smart communications objects belong to group with long term relations of size scaling from very few (objects belonging to unique person), to hundred of devices. Those objects hosted by people are organized in a social group with common rules. Those communication devices participate to spontaneous and self-organized networks with encountered other mobiles devices and with an always more and more intelligent environment. Contexts of interaction range

from access to authenticated servers or to unexpected interactions with unknown devices.

Such an environment introduces new security challenges that are not adequately addressed by existing security models. Without centralized organization, security turns in a risk management problem where specific trust model and associated cryptographic techniques are required. Each device needs to carry self-contained information and methods to be able to make fully autonomous trust decisions.

In human interaction, trust is a *continuous variable* that is compared to the aim of the interaction to evaluate the risk of the operation. Our computational model of trust captures trust dynamics of the human society. As mentioned in [6], trust is subjective and individual as suggested according to Gambetta's definition [9]: "Trust is the subjective probability by which an individual, Alice, expects that another individual, Bob, performs a given action on which its welfare depends". Trust also depends on stable groups such as family, friends or colleagues at work defining subjective trusted communities.

The aim of this paper is to describe and analyze a complete trust model dedicated to smart mobile communicating devices. Our proposal holds all the desired properties for a distributed trust framework. First of all, the proposed framework *derives from human social system* in order to be socially accepted. Human evaluation of trust is a complex system and is difficult to mimic in a computational model. Therefore, we know that this evaluation is a combination of past experiences and external information that can be simplified as recommendation information. Human evaluation of trust is also depending on the *context of interaction*.

In our trust model, past experiences and recommendations are aggregated in the notion of *history of past interactions* (as proposed in [6]) which are protected by cryptographic material. Context may be derived by collecting past history of objects in the environment. In our model, the acting peer tries to forge a direct experience with the target party using the content of their own histories. We avoid the trust dissemination, each entity is viewed as an autonomous device and the trust evaluation is based only upon selfish evaluation of common trustworthy nodes. The trust level is then computed only after successful transactions corresponding with a positive reputation mechanism as described in [18]. Our proposal reduces the complexity of the decision-making process by providing *proved data* that can be the foundation of the final decision.

Besides already presented properties, our trust model is highly adaptable and parameters can be set to correspond to various model of communities, each with its own trust policy. Our model is also *robust to classical security attacks* like Sybil and man in the middle attacks. And last, our proposal can be fitted in a *light weight* decision module, both in term of required computing capability and bandwidth requirement.

This paper is organized as follows: section 2 presents relevant approaches concerning trust and trust management framework. Section 3 specifies the proposed history based trust approach and provides an overview of our protocol

(already described in [8]) with a dedicated security analysis. Section 4 gives the trust metric and explains the impact of the parameters associated with the metric. Section 5 shows using intensive simulations the effect of the parameters on the dynamics of groups using our model.

2 Related Work

According to [18], trust management systems are classified into three categories: credential and policy-based trust management, reputation-based trust management, and social network-based trust management. This approach depends on the way trust relationships between nodes are established and evaluated. In credential and policy-based trust management system [2–4], a node uses credential verification to establish a trust relationship with other nodes. The concept of trust management is limited to verifying credentials and restricting access to resources according to application-defined policies: they aim to enable access control [10]. A resource-owner provides a requesting node access to a restricted resource only if it can verify the credentials of the requesting node either directly or through a web of trust [11]. This is useful by itself only for those applications that assume implicit trust in the resource owner. Since these policy-based access control trust mechanisms do not incorporate the need of the requesting peer to establish trust in the resource-owner, they by themselves do not provide a complete generic trust management solution for all decentralized applications. Reputation-based trust management systems on the other hand provide a mechanism by which a node requesting a resource may evaluate its trust in the reliability of the resource and the node providing the resource. Trust value assigned to a trust relationship is a function of the combination of the nodes global reputation and the evaluating nodes perception of that node. The third kind of trust management systems, in addition, utilize social relationships between nodes when computing trust and reputation values. In particular, they analyze the social network which represents the relationships existing within a community and they form conclusions about nodes reputations based on different aspects of the social network. Examples of such trust management systems include Regret [16, 17] that identifies groups using the social network, and NodeRanking [14] that identifies experts using the social network.

Ambient networks are environments where only a distributed reputation system is allowed [13]: there is neither centralized functions nor central location for submitting the ratings or for obtaining the reputation scores of nodes. Each participant simply records his opinion deduced from his own experience about another party. A node, in order to protect itself from potential malicious nodes, trusts only information which is obtained locally: a communication protocol allows all participants to obtain ratings from each other. The reputation of a target party is computed by a specific agent with the help of requested ratings and possibly from other sources of information. Of course, proper experiences with a target party carry a weight higher than the received ratings. But it is

not always the case and the main difficulty is thus to find the distributed stores which deliver these specific ratings considering that the trust data is disseminated in numerous stores. Nevertheless, this information is easily provided on request for a relying party.

3 Our trust management framework

3.1 Our model

General Overview Current trust management systems suppose that most of the encountered terminals are honest so that the number of malicious information is not enough important to mislead the trust decision process. If this assumption is true in some general context, it does no longer hold for personal communicating devices in ambient and intelligent environment. Except in the case of direct interactions between each mobile device such as in personal experiences, all knowledge comes from uncertified devices. Moreover, the availability of this information is limited because of the restricted size of storage of these mobiles.

Our trust management framework is thus designed for decentralized environment where only partial information is available. Trust is evaluated and derived from the following types of information: past personal experiences, encountered device recommendations, and contextual information such as the moment and the place where the interaction takes place. A history based approach (as in [6, 12]) is used in combination with some cryptographic materials: in case of a successful interaction, each involved node stores a *history element* signed by both parties. The number of interactions with a node called *intensity* of interaction, is also stored. The semantics of a history element is important but this is out of the scope of this paper (see [8]). Each node also carries a *blacklist* which takes into account the untrustworthy nodes. This situation may occur because these nodes were dishonest during several interactions or the service did not repeatedly proceed properly. The full management policy of this blacklist is also out of the scope of this paper.

Thus, a history implies that trust decision process is based on the validity of exchanged information since it not only relies on the honesty of the transmitted information, but also it depends on fully *certified* data: the mobiles are thus incited to be honest regarding transmitted information. In the case of multiple interactions with the same node, the device has to keep only the last proof of interaction to lower the volume of recorded data and the computing overhead.

To sum up our proposition, a node A evaluates the trustworthiness in a node B using only local information: its history H_A, the intensity $I_A(B)$ of the relation with B, its own blacklist BL_A, and the history H_B transmitted by B. With the help of cryptographic algorithms (see section 3.2), node A can check the validity of any history element in H_B as soon as it has the knowledge of the public identity of the involved nodes. As explained later, the verification

is restricted to $H_A \cap H_B$ which corresponds to the known common devices between A and B. Conserve a history element relating A and B means *node A recommends node B* but unlike classical recommendation framework, this assumption can be verified.

The lifetime approach In an environment where exists neither a central regulating entity nor authorizing accreditations or the revocation of objects, a possibility is to let make time: the data elements are automatically revoked after their lifespans expire [15]. A temporal semantics can easily be added to a history element if both nodes agree on a creation and an expiration date. This information is simply concatenated with existent data before the signature. Nevertheless, nothing guarantees that the both entities will choose correct values for this information: the reality may be different (dishonest nodes or malfunction). But there is no real benefit to cheat on these values. Indeed, each entity may filter a received history element according to its local trust policy: an element can be rejected if its creation date is too old, its validity period is considered to be abnormally long although being still valid or if its lifespan is of course expired. No information having an infinite lifespan in the system is guaranteed by this timestamp.

Identity and impregnation It is of course impossible in absolute to avoid the compromise of a node either by a technical action (hacking) or by a social engineering attack (stealing of password, ...). Consequently, an attacker who compromises a mobile has a direct access on the history elements present on this device. This problem is addressed in our model through the impregnation of a device with an identity.

Identity is set at the birth of the mobile device and is the result of a collaboration between the node (or the owner) and a special device called *imprinting station*. Although this imprinting station implements the trust model but it is not certified by any authority. Each imprinting station defines a domain of security which corresponds to a dedicated social group. A domain may contain a large group of mobile devices or just a single smart mobile device embedding its own imprinting station.

At the end, a node A can easily check that an encounter B either belong to the same community or not by verifying their respective imprinted signatures. Then, we could define the $C(A, B)$ parameter which is a boolean value assigned to true if and only if A and B belong to the same community. We call this value the *community parameter*. Depending of the social model underlying a community, this parameters can be included or not in the trust function.

To mitigate the impact of compromised nodes, the identity has also a lifespan. Before its expiration time, a node needs to be re-initiated by its imprinting station in order to update its new identity lifespan. A cryptographic link is created between two consecutive identities ID_1 and ID_2 (as explained in the section 3.2). While there exists some elements that are non expired or signed with the older identity, this identity is yet presented to check previous history elements. The new identity is used to sign new history elements.

3.2 A detailed approach of our protocol

An entity of our model is equipped at least with a *cryptographic package* what will make it compatible de facto with any other entity of our model, i.e. other objects which implicitly accept the model. When an object received this package and the initial parameters, it can then initiate sessions of communication by the means of the CHE protocol (detailed in [8]. If a session is accepted, the two involved nodes estimate, considering their security policies, that they can trust each other during this interaction.

Starting from an empty history, a node records all the successful interactions made with other nodes in order to support the future spontaneous interactions. To prove the past interactions, it creates with each met node an element of history related to their respective identities and signed by the two parts. Before any interactions, nodes must build a *trust germ*, by counting the number of common nodes they have in their history, or by manually forcing the relation: this is the *bootstrap* phase of our model. If the number of common interactions is sufficient (greater than a threshold p which is a function of the size n of the community and the maximum size H_{max} of the history), they can then interact.

The initial seed of trust Each device receives an initial *trust germ* from its *imprinting station*. It is composed by the following information: an identifier ID_u chosen by the device owner (eMail adress or IP address or just a simple name or pseudonym) supposed to be unique within the security domain built by this imprinting station, an identity which is obtained from this identifier by concatenating it with a date d and a lifespan T ($\mathrm{ID} = \mathrm{ID}_u||d||T$), a first pair of private/public key $(S_{\mathrm{ID}}, Q_{\mathrm{ID}})$ for cipher operations, a second pair of keys $(S^S_{\mathrm{ID}}, Q^S_{\mathrm{ID}})$ for the signature and a set representing all the public parameters of the elliptic curves required along computations:

$$\mathsf{Params}\text{: } \Omega := \langle \mathbb{F}_p, a, b, P, h, G_1, G_2, e, H_1, H_2, H'_1, H'_2; P_{pub,\Omega} \rangle$$

where: a and b are the parameters of a particular elliptic curve $y^2 = x^3 + ax + b$ on $\mathbb{F}_p$; P, a particular point of this curve of prime order q; h, the cofactor defined as $h = \#E(\mathbb{F}_p)/q$; G_1, is a first additive cyclic group of prime order q built using the P point; G_2, a multiplicative cyclic group of the same order; e, a bilinear pairing from $G_1 \times G_1$ to G_2; $H_1 : \{0,1\}^* \to G_1^*$ and $H_2 : G_2 \to \{0,1\}^n$, two map-to-point hash functions required for the Boneh-Franklin's Identity Based Encrytion (BF-IBE) (see [5] for more details); and $H'_1 : \{0,1\}^* \times G_1 \to G_1$ and $H'_2 : \{0,1\}^* \times G_1 \to \mathbb{Z}_q$, two hash functions required for the Chen-Zhang-Kim IBS signature scheme (CZK-IBS) (see [7] for more details). Notice that the node public keys are directly derived from their identities due to the use of Identity-Based cryptosystems.

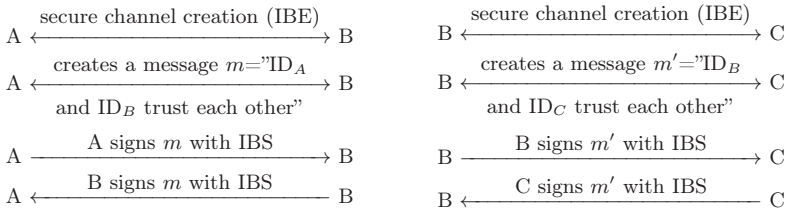
Another important point is that each smart device shares the same following cryptographic algorithms and protocols downloaded from the imprinting station: a fingerprint algorithm, a signature algorithm, a zero-knowledge protocol, a protocol to construct secure channel and the public parameters.

Ω-values are the domain identifier values provided to each node imprinted by the same imprinting station. Every imprinting station possesses the same Ω-values except $P_{pub,\Omega} = sP$ varying along the parameter s, the master key of a station. This value depends on each station and must be absolutely kept secret by it. None of these imprinting stations is supposed to be certified by any authority. Moreover, an independent mobile imprinting itself may be its own standalone security domain. The only values that each smart device has to keep secret is S_{ID} and S^S_{ID} as usually in cryptosystems.

Notice that if a first identity is $\mathrm{ID}_1 = (\mathrm{ID}_u||d_1||T_1)$ where ID_u represents the name or a pseudonym, d_1 a date and T_1 a lifespan, this identity allows to generate the first corresponding key pairs. Then, the updated identity ID_2 is equal to $\mathrm{ID}_2 = ((\mathrm{ID}_u||d_2||T_2)||MAC((\mathrm{ID}_1||\mathrm{ID}_u||d_2||T_2), P_{pub,\Omega}))$ where d_2 represents a second date, T_2 another lifespan and MAC is a MAC algorithm. And so on, the next identities are created using the same operations, generating a MAC chain.

The reciprocal trust Once the initialization phase is done, a node may interact with other nodes without any contacts with its imprinting station. This forms a second phase in the protocol.

The first step of our protocol supposes that both entities Alice and Bob have already interacted at least once and have built a trust bond: this is a message m signed by Bob that Alice publishes in the public part of her history $(m, sign_B(m))$ while Bob publishes $(m, sign_A(m))$ in its own history. This bond could be created by forcing by the hand the beginning interaction as in a Bluetooth like system if the number of common elements of their history were insufficient. Let us note that if Alice and Bob have already met and if this new interaction is successfull, they just have to modify the respective values of the intensity and to rebuild a new history element to replace the old one because it contains a timestamp. Suppose now that in the same way Bob and Charlie have built a secure channel to exchange a common message of mutual trust m'.



The second step of our protocol describes a trust bond establishment using history contents between two entities (here Alice and Charlie) that have never met. Thus, when Alice meets Charlie for the first time, they exchange the concatenation of all the public keys Q_{ID} contained in their history. Once this first exchange carried out, Alice and Charlie realize that they have both met

before Bob and want to mutually prove this common meeting. Charlie, first, proves to Alice that Bob trusts him using the message m'. Alice could verify the contents of m' because she knows Bob's public keys from her own previous meeting.

$$\begin{array}{rcl} \text{A} & \xrightarrow{\text{did you meet Bob before ?}} & \text{C} \\ \text{A} & \xleftarrow{(m', sign_{S_B^{\widetilde{S}}}(m'))} & \text{C} \end{array}$$

verifies m'

The reciprocal process will be then repeated by Alice.

3.3 Security analysis

Security requirements The following traditional cryptographic properties are guaranteed by our protocol: an offline authentication (users performs each other a weak authentication using the IBE scheme and as Charlie knows the Bob's public keys, he could authenticate his signature), integrity is guaranteed by the hash function used in the IBS scheme as in the classical case of a certificate, confidentiality is guaranteed by the use of the cryptographic IDs. Those IDs also permit to guarantee that the first phase of our protocol was correctly done. The secure channel built at the beginning of the exchange in the first phase also prevents a man-in-the-middle attack.

The user could preserve its anonymity because he is free to choose his own pseudonyms according the context and could have several pseudonyms distributed by different imprinting stations. Those pseudonyms are certified through the used identity-based schemes and they preserve the real identity of their owner, even if his meetings when he acts in the network are known with other peers with pseudonyms. Moreover, each identity defines its own history and all the pseudonyms are certified, thus tackling "Sybil attacks". Our model also guarantees the non-repudiation: each user is preventing from denying previous meetings or actions. Revocation is also possible using the timestamp linked with an ID and included in the key pairs (as previously described in 3.2).

Classical attacks As mentioned in [8] and due to the use of the IBE-scheme, the well known key escrow drawback is inherently present in our protocol. We then suppose that all the imprinting stations must be trusted entities. Otherwise, they can read and send messages instead of nodes. However, the signature scheme used here prevents such an attack from happening because the signature key pair generated is unknown from the imprinting station.

Our trust management framework is a cross-domain protocol: two nodes, not belonging to the same domain (or to the same imprinting station) could nevertheless interact by comparing the contents of their respective histories once

they exchange the public key of their security domains (we suppose here that all the other parameters are the same).

Our protocol also guarantees the non-transferability of the history because only the knowledge of the secret keys allows to use the content of the history (the secure channel initially built prevents the use of the history elements). Then, stolen identities or pseudonyms or histories could not be useful.

Sybil-like attacks A node or a user could using our protocol forges several identities or pseudonyms from the same or different imprinting stations and then uses them in Sybil-like attacks. However, in our model, one identity or pseudonym could only be linked with a particular history.

For example, suppose that an attacker (Eve) wants to attack a single entity Alice, then she creates first several pseudonyms $S_1, \cdots, S_n$. Alice asks her a particular service, they realize that they have enough common history elements to interact. Suppose now that Eve does not provide the corresponding service to Alice with her S_1 pseudonym, Alice then decides to blacklist the S_1 Eve's pseudonym. So, Eve must use an other pseudonym, S_2 for example, if she wants to interact and attack Alice again. To manage this operation, she must build an other time a sufficient number of history elements common with Alice. Even if, she knows the pseudonyms of nodes to meet again with her second pseudonym, she must play an active and positive role inside the "Alice's friends". The attack using several pseudonyms is then very expensive in our case and requires lots of social engineering.

Clone attacks As mentioned in [8], a major attack against our model is the clone one where Alice clones herself with some other terminals. Those clones with exactly the same keys could build a very strong history and have lots of recorded elements and could interact more easily than the others. Therefore, AliceŠs cloned devices could be carried by different persons visiting different places in order to have different histories. This is not considered by us as a major risk since it is a social engineering attack which is difficult to conduct as well as difficult to surround by cryptographic methods.

4 General Context of our analysis

Having presented the basic block of our trust management framework and having discussed its security requirements, we then describe the general context of our framework main processes: how a node A really computes the trust value concerning the node B, supposing that the node A is the service provider - the trusty - whereas the node B is the trustor.

First, we give a general overview of our notations and then we introduce a function which rates the trustfulness between each node.

4.1 General context

For sake of simplicity, we consider a unique community which is characterized by its *size* n. The community dynamics depends of the *interaction rate* of each individual node, i.e. the average number of interactions by unit of time. The other parameters are H_{max} the maximal size of a history which is the same for all nodes and BL_{max} the maximal size of the blacklist which is also the same for all nodes. The node A stores its trusted nodes in its history H_A and reciprocally, it stores untrustworthy nodes in its blacklist BL_A. Hence, to a trusted node B corresponds an element $h_A(B)$ in the history H_A. In addition, each element is tagged with two fields: the first one ,denoted by $I_A(B)$, represents the intensity of the relation with B, the second, denoted by $U_A(B)$, represents the *utility* of B, i.e. the usefulness of the node B with respect to the node A. This last notion is related to the number of times this element contributes in the computation of common elements.

In a more general framework, with different communities, trust policy could be different according to either an interaction takes place with a member of its community or not, taking into account the $C(A, B)$ *community parameter*. More deeply, the internal structure of a community could also modify the trust policy: for instance, through the social degree of the community, initial trust may be total: each node is valid and active in the community (for example for objects belonging to a same family). On the contrary, the initial trust may be partial or even non-existent if the social degree of these communities is loose (for example for objects belonging to members of a national sporting federation with several thousand of members). In this case, the weight given to the *community parameter* could no more be the same and the behavior of a mobile in such a community depends essentially of its own experiences through its history.

4.2 Trust function

Direct trust value We first introduce the main element of our trust function. Suppose now that A and B are two nodes belonging or not to the same community. The main element of trust in our model is the threshold of common history elements. To compute this value, we need to introduce the direct trust value:

$$d(A, B) = \alpha|H_A \cap H_B| + (\alpha - 1)|BL_A \cap H_B|$$

where α varies in the interval $[0, 1]$.

This coefficient indicates the weight of the number of common elements $|H_A \cap H_B|$ versus the number of untrustworthy nodes of A that B considers trustfulness. This value obviously admits negative value for some values of $|BL_A \cap H_B|$, but we consider that if its value exceeds a positive threshold p, then the associated value $T(A, B)$, representing the direct trust level, is equal to one, otherwise it is equal to 0. The parameter p defines thus the threshold for a direct trust.

General trust function A trust value is also context-dependent, so we need to combine the direct trust value with other values given by the context of the interaction, limited here to the *intensity* of the relation between the two involved nodes and to the *community parameter*. We then compute the general trust function about the node B viewed by the node A using the following formula:

$$TF_A(B) = \frac{\beta_A T(A,B) + \gamma_A \frac{I_A(B)}{I_{max}(A)} + \delta_A C(A,B)}{\beta_A + \gamma_A + \delta_A}$$

with $T(A,B) = 1$ if $d(A,B) \geq p$, 0 otherwise and with $C(A,B) = 1$ if $\Omega_A = \Omega_B$, 0 otherwise; where β_A, γ_A and δ_A are three parameters that belong to $[0,1]$; and where $I_{max}(A)$ represents the maximal intensity admitted by A. Then, The general trust function gives a trust notation that belongs to $[0,1]$. According this value and the trust threshold t_{ID} defined by each node, the involved nodes could decide to interact or not.

The β, γ and δ values represent the weights of each parameter we want to take into account. They depend on the local trust policy of the node. For example, a node will prefer, if it has never met a node C (then, the corresponding $I_A(C)$ value is equal to 0) takes into account the number of encountered nodes represented by the β parameter than the community one (they belong to the same tennis club). More precisely, the δ parameter represents the degree of structure of a community and will depend on the type of the community.

5 Experiments and Simulation results

We aim here to propose a set of rules for the management of a group by evaluating the various parameters in order to make the mobiles as autonomous as possible: we seek for instance to lower the duration from which the dynamic process takes the top compared to the bootstrap phase (Fig. 1) by adjusting the different parameters (the maximum history size H_{max}, the metric threshold p,...).

In this section, the presented simulations only compute the $d(A,B)$ parameter, the most important one, considering the other ones as some bonus of interactions. The presented simulations don't take into account the blacklist process. However, this process can be relatively simply added in the management. This can be seen like another instantiation of the model. Let us recall also that two nodes having interacted jointly several times keep only one element of history: this element is updated each time as necessary. For needs of the performed simulations, two types of network were considered: The first type was built upon a uniform random distribution which selects the pairs of interacting nodes, while for the second type, the pairs of nodes are picked with respect to a power law distribution.

The bootstrap phase The bootstrap phase of a node is very important in our proposition and requires the intervention of its owner. At initial step, the

history of the node A is empty of trusted elements. And thus, the metric above is useless since no terms can be evaluate. Hence, each trusted element $h_A(B)$ (resp. each blacklisted element B) which is added by the user in the history (resp. in the blacklist) of A has a great impact in the dynamics of this node.

It is also important to notice that this phase implicitly has an impact on the defense mechanism of a group: a group may protect itself by imposing a strong *social activity* to users belonging to another groups. A malevolent user which has compromised an object, must act inside the group in order to avoid losing the benefit of his situation. The situation is quite the same for a benevolent user who leaves for a long time his group: if he does not anymore maintain his relations and wants to be reintegrated, he must undertake one more time the *bootstrap* phase. This fact can be seen like a disadvantage of our protocol, nevertheless, initiate a *bootstrap* phase is easier for an authorized user than for an attacker.

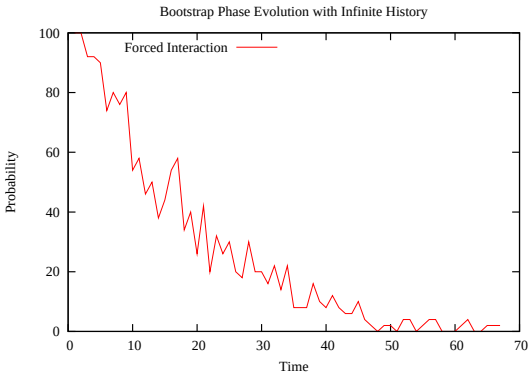


Fig. 1. Evolution of forced interactions (expressed in percentage) by section of 50 steps of time for a community of size $n = 100$ nodes considering a history with an infinite size.

Eviction policy of the history A major element of the policy management of a community is the rule to apply for incorporating a new element in a saturated history. Indeed, the size of the history is necessarily limited for a mobile with small resources. We plan here to set up two modes of replacement. The first mode, which is denoted by FIFO (*First Input First Output*), removes the *oldest* element out of a considered node history: the first withdrawn element has the oldest date. Such a policy allows thus to make disappear the mobiles which are no longer active in the community. The second mode, which is denoted by LUFO (*Less Used First Output*), withdraws the useless elements which appear in the computation of common elements. To measure the importance of each

history element, we take into account the number of times this element is used to compute the number of common elements: each common element between the two parts i and j is credited with one point, this corresponds to the value $U_i(j)$ that represents the *utility* of an element. A history element having the lowest points account is purged and replaced by the partner of the interaction if the current interaction succeeds. We focus on the study of the probability - expressed in percentage - that two distinct nodes have a sufficient number of common elements in their respective histories at time t. This probability is: $P(t) = \frac{2}{n(n-1)} \sum_{i \neq j} T(i,j)$ with $n(n-1)/2$ corresponding to the total number of distinct nodes pairs and $T(i,j)$ is the boolean value previously defined without taking into account the blacklist process.

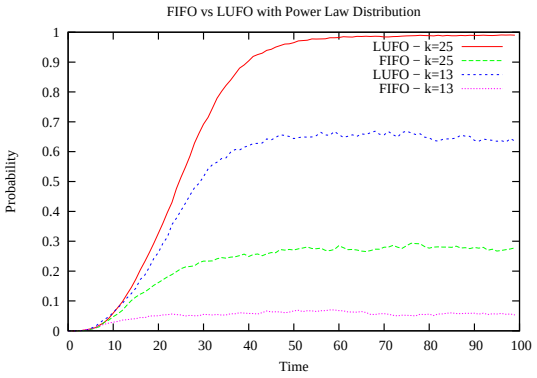


Fig. 2. Evolution of the P probability during time t for several k values (history size) according the eviction mode used (LUFO or FIFO). Threshold: $p = 3$ for $n = 100$ nodes).

We have also computed such a probability by using the birthday paradox for several values of H_{max} ranging from $0,5 \times n/\ln(n)$ to $1,2 \times n/\ln(n)$, whereas the threshold p ranging from $0,5 \times \sqrt{n/\ln(n)}$ to $1,5 \times \sqrt{n/\ln(n)}$. On the one hand, a quick analysis shows that the obtained computation results are not really different as well as the case of a random distribution as the case of a power law distribution (however, with a light profit for this last). On the other hand, there is a great difference in behavior of the model according to the mode of replacement used (LUFO or FIFO) as shown in Figure 2.

In conclusion, this analysis shows as results that the LUFO mode is more efficient to keep the cohesion of a regular interacting nodes group than the FIFO mode. Indeed, the FIFO policy does not take into account the importance of the history elements. On the contrary, if we only keep the most active and useful elements, the chances to find them in other histories are increased. Their

number is thus often greater than the threshold p. In consequence, the choice of an eviction policy is very clear: the LUFO mode using the $U_i(j)$ value is opted. In addition, fixing a threshold at 3 or 4 is reasonable for communities of 100 or 200 nodes. Beyond, the protocol would require too many user interventions to be viable. Another information from this analysis is the choice of a power law distribution versus a uniform random distribution to describe the behavior of the nodes activities in the community is negligible.

Impact of blacklist As we announced in the description of the model, the use of a blacklist is highly desirable. In a fully distributed enviroment, adding a node in a blacklist is the only possible sanction if its behavior is considered to be incorrect. It corresponds to a perfectly identified social behavior. In a risk evaluation approach, it can appear logical to penalize the nodes which present recommendation coming from nodes which are blacklisted.

The disadvantage of the blacklist policy is to prohibit some interaction with honest nodes only because some element of their history have been locally blacklisted. There is thus a balance to find between a very strict policy which will have a very negative impact on the whole of the community and permissive policy which will imply a too important taking risk.

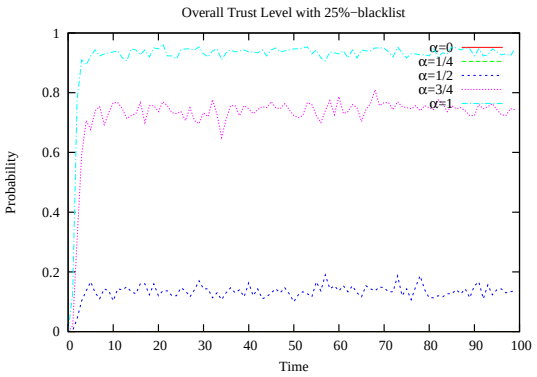


Fig. 3. This figure describes overall trust level of a 25-nodes community according to whether the blacklist elements are considered or not. Each node has a fixed history size $BL_{max} + H_{max} = 10$.

The figure 3 shows evolution of the overall trust expressed in percentage along the time for a 25-nodes community. We observe that as the coefficient α decreases as does overall trust. Such a behavior is what expected. This observation could be extended for all values of α: for its small values, the dynamics of the system is stopped due to the higher importance of blacklisted elements than

common elements in the history of each node. In contrast, for values around one, the dynamics of the system is not impacted by the presence of these blacklisted elements.

context awareness As we announced at the beginning of this paper, our model allows the introduction of context awareness evaluation. A node A can carry out easily a passive listening of the elements of the histories H_B and H_C exchanged by two nodes which apply our protocol in its radio range. The node can compute the cardinality of both $H_A \cap H_B$ and $H_A \cap H_C$. By carrying out regular listening, the node can evaluate the social situation in which it is embedded. If the cardinality of intersected histories is always high the node could consider that it acts in a well known environment and may adapt its security policy to this context. The use of our protocol confers on the model two major advantages. First of all, context awareness evaluation is made of data that can by checked cryptographically. Secondly, with only some listening nodes obtain a large list of history elements what enables us to acquire this information very quickly. This second point is important to reduce the cost of such a listing and of the evaluation process.

Let us consider a node A belonging to a community C and 3 contexts where the proportion C nodes of surrounding A are respectively 80%, 50% and 10%. The objectif of a contexte awareness evaluation for node A is to detect as quick as possible in which A is really embedded. As we explained at the beginning of this section, with a bounded history, even while being in its own community, the intersection of history is not always sufficient for spontaneous interaction. This detection can be established by the gap in the ratio of known nodes over unknown nodes. Known nodes mean here those stored in its history. This ratio may be accurate with few samples. This could be proved analytically using again the birthday paradox.

6 Conclusion

We have proposed a distributed framework that produces trust assessments based on proved direct experience. Our cross-domain scheme supports a weak authentication process, user anonymity and resists to lots of attacks, especially the Sybil-like one. From this basis, we have designed a trust notation that takes into account a local blacklist process and that is context awareness. Finally, we have conducted experiments which show that this framework is suitable for large communities, the bootstrap phase being not an obstacle and that the blacklist process well prevents trusted nodes from the malicious behavior of some peers. As part of future work, we will investigate the dynamics of our model behavior for some special social cases.

References

1. *The First International Joint Conference on Autonomous Agents & Multiagent Systems, AAMAS 2002, July 15-19, 2002, Bologna, Italy, Proceedings*. ACM, 2002.
2. Matt Blaze, Joan Feigenbaum, John Ioannidis, and Angelos D. Keromytis. The KeyNote Trust-Management System Version 2 - RFC 2704. RFC 2704, Available from `http://www.faqs.org/rfcs/rfc2704.html`, September 1999.
3. Matt Blaze, Joan Feigenbaum, and Angelos D. Keromytis. The role of trust management in distributed systems security. In Jan Vitek and Christian Damsgaard Jensen, editors, *Secure Internet Programming*, volume 1603 of *Lecture Notes in Computer Science*, pages 185–210. Springer, 1999.
4. Matt Blaze, Joan Feigenbaum, and Jack Lacy. Decentralized trust management. In *IEEE Symposium on Security and Privacy*, pages 164–173. IEEE Computer Society, 1996.
5. Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. In *Advances in Cryptology - Crypto'2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer, 2001.
6. Licia Capra. Engineering human trust in mobile system collaborations. In Richard N. Taylor and Matthew B. Dwyer, editors, *SIGSOFT FSE*, pages 107–116. ACM, 2004.
7. Xiofeng Chen, Fangguo Zhang, and Kwandjo Kim. A new ID-based group signature scheme from bilinear pairings. In *Information Security Applications, 4th International Workshop - WISA'03*, volume 2908 of *Lecture Notes in Computer Science*, pages 585–592. Springer-Verlag, 2003.
8. Samuel Galice, Marine Minier, John Mullins, and Stéphane Ubéda. Cryptographic protocol to establish trusted history of interactions. In *Third European Workshop on Security and Privacy in Ad hoc and Sensor Networks*, page LNCS 4357, september 2006.
9. Diego Gambetta. Can we trust trust? In Diego Gambetta, editor, *Trust: Making and Breaking Cooperative Relatioins*, chapter 13, pages 213–237. Published Online, 2000.
10. Tyrone Grandison and Morris Sloman. A survey of trust in internet applications. *IEEE Communications Surveys and Tutorials*, 3(4), 2000.
11. Rohit Khare and Adam Rifkin. Weaving a Web of trust. issue of the World Wide Web Journal (Volume 2, Number 3, Pages 77-112), Summer 1997.
12. Véronique Legrand, Dana Hooshmand, and Stéphane Ubéda. Trusted ambient community for self-securing hybrid networks. Research Report 5027, INRIA, 2003.
13. Filip Perich, Jeffrey Undercoffer, Lalana Kagal, Anupam Joshi, Timothy Finin, and Yelena Yesha. In reputation we believe: Query processing in mobile ad-hoc networks. *mobiquitous*, 00:326–334, 2004.
14. Josep M. Pujol, Ramon Sangüesa, and Jordi Delgado. Extracting reputation in multi agent systems by means of social network topology. In *AAMAS* [1], pages 467–474.
15. Daniele Quercia, Stephen Hailes, and Licia Capra. Tata: Towards anonymous trusted authentication. In Ketil Stølen, William H. Winsborough, Fabio Martinelli, and Fabio Massacci, editors, *iTrust*, volume 3986 of *Lecture Notes in Computer Science*, pages 313–323. Springer, 2006.
16. Jordi Sabater and Carles Sierra. Regret: reputation in gregarious societies. In *Agents*, pages 194–195, 2001.

17. Jordi Sabater and Carles Sierra. Reputation and social network analysis in multiagent systems. In *AAMAS* [1], pages 475–482.
18. Girish Suryanarayana and Richard N. Taylor. A survey of trust management and resource discovery technologies in peer-to-peer applications.

Detecting Wormhole Attacks in Wireless Networks Using Local Neighborhood Information

Wassim Znaidi, Marine Minier and Jean-Philippe Babau
CITI / INSA-Lyon - F-69621 Villeurbanne
Email: {wassim.znaidi,marine.minier,jean-philippe.babau}@insa-lyon.fr

***Abstract*—*Wormhole attacks* enable an attacker with limited resources and no cryptographic material to disrupt wireless networks. In a wormhole attack, an attacker records packets (or bits) at one location in the network, tunnels them (possibly selectively) to another location and retransmits them there into the network. In this paper, we present an algorithm for detecting and thus defending against wormhole attacks in wireless multi-hop networks. This algorithm uses only *local and neighborhood information* without requiring clock synchronization, location information or dedicated hardware. Moreover, the algorithm is independent of wireless communication models. We present simulation results for grid-like topologies and for random topologies and show that the algorithm is able to detect wormhole attacks in all cases whereas the number of false alarms (false detections) decreases rapidly if the network is sufficiently dense.**

I. Introduction

Wireless ad-hoc and sensor networks are emerging to solve challenging problems. They generally use a wireless radio communication channel allowing rapid deployment and low-cost operation and they are automatically self-configuring and self-maintaining. However, they are most of the time deployed in an open and uncontrolled environment that requires secure communication and routing. In particular, emergency response operations, military or police networks need secure communications and real-time feedback. Due to their wireless and collaborative nature, those networks are vulnerable to several attacks. In most wireless networks, an attacker can easily *scramble* communications, *inject* false packets and also *eavesdrop* on communication and *replay* the recorded packets.

In this paper, we focus on a particularly dangerous attack called the *wormhole attack* as described in [1] for example. Here, an adversary installs a dedicated connection between two distant points by a variety of means (e.g. a network cable, a long-range wireless transmission in a different band,...) with a high quality and a low latency. Messages received by one wormhole endpoint node are retransmitted at the other endpoint node. Using this wormhole link, attackers can exploit wormholes to build bogus route information and to attract the network traffic. They obtain a very powerful position to launch many attacks such as Denial-of-Service attacks, routing loops, etc.

In this paper, we propose a localized algorithm for detecting wormhole attacks which is purely based on local neighborhood information. We assume that this information is collected by various upper layer neighbor discovery protocols. No additional hardware artifact is required making the approach generic. The detection algorithm essentially looks at local structures that must be observed when there is no wormhole in the network and that disappear when a wormhole link is present in the network. We demonstrate the capabilities of our algorithm using several topologies in our simulations.

Section II describes the wormhole attack and presents the related work that develops countermeasures for wormhole attacks. In Section III, we present our assumptions and our algorithm. Section IV provides simulation results of our algorithm and Section V concludes.

II. Wormhole Attacks and related work

In this section, we give more details about the threats induced by the wormhole attacks and describe the known countermeasures developed against wormhole attacks.

A. Wormhole attack

In a wormhole attack, an attacker receives packets at one point in the network, "'tunnels" them to another point in the network via an out-of-band connection and replays them at another location in the network [2]. In Figure 1, the attacker replays packets received by X to Y and vice versa. As a result of this attack, packets sent by neighbors of X will be heard through the wormhole by the neighbors of Y. Thus, node A for example could be convinced that node B is its 3-hop neighbor and vice versa. For example, node A could believe that node B is reachable in 3-hop and thus node C could then transmit its packets for B via A. So, if the wormhole is placed carefully by an attacker and is long enough, it can create many false routes (not only for the wormhole nodes but beyond) and have dangerous effects on routing protocols since it influences the topology construction.

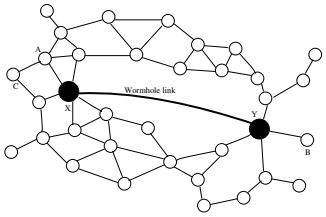


Fig. 1: Wormhole attack. The wormhole nodes X and Y are connected through a wormhole link.

The wormhole attack is particularly dangerous against many ad hoc network routing protocols such as on-demand routing protocols ([3] and [4]) and their secure versions ([5])

because ROUTE REQUEST messages could be tunneled directly to nodes near the destination target node through the wormhole link. Moreover, this ROUTE REQUEST message arrives earlier than the others due to the high quality of the wormhole link. This attack prevents any other routes from being discovered.

After having attracted a lot of data traffic through the wormhole, the attacker can selectively discard messages or modify them to create partial or permanent DoS attacks, selective forwarding attacks, create a sinkhole in a sensor network if a wormhole node is located near the base station, etc.

Moreover, the adversary does not need to decrypt data (if an encryption system is used) or wait to receive all the packets because he can operate on a bit-by-bit basis which makes the wormhole attack more difficult to detect using for example timing analysis. As noticed in [6], using wormholes, an attacker can also break protocols based upon geographic proximity or connectivity-based localization algorithms in ad hoc or sensor networks where GPS could not be considered.

B. Related Work

As the problem of security in ad-hoc network has become a major concern, several secure protocols have been proposed in the literature to detect wormhole attacks. Unfortunately, many of the proposals cannot easily be applied to generic wireless networks because they use dedicated hardware or location information not always easy to obtain.

In [2] Hu and al. use packet leashes to detect wormhole attacks. Packet leashes contain geographical or temporal information to bound the distance or the lifetime of an end-to-end transmitted packet to restrict its travel to the destination. The sender includes a timestamp or (some) localization information in the message and the receiver checks if the received packets are in "legal" time or in legal distance. This method requires tightly synchronized clocks and a precise location knowledge like GPS, that can be hard to obtain for small devices.

In [7], the authors propose the MAD protocol (Mutual Authenticated Distance-bounding). This approach is based on the concept of distance-bounding without using any GPS or synchronized clocks. One node has to respond instantaneously to a series of bit exchanges sent by a second node who compute the distance between them using timing properties. However, the MAD protocol needs a special hardware radio to switch very quickly between send and receive mode.

In [8], the authors propose to use directional antennas to detect wormhole attacks. Each node maintains a neighbor list using a neighbor discovery protocol based on a legal verifier that provides correct directional information. However, the use of directional antennas limits the use of this protocol.

In [9], the multi-dimensional scaling (MDS) algorithm which builds a virtual layout of the network is proposed. Each node reports its list of neighbors and their estimated distances to their neighbors. Then the algorithm tries to determine the possible positions of each node in such a way that the constraints induced by the connectivity and the distance estimation data are respected. A distortion in the layout will appear if the wormhole attack exists. Another solution [10] consists in using statistical tests (such as the χ^2-test) applied on the connectivity graph constructed by the base station. The presence of a wormhole would increase the number of edges in the graph constructed by the base station using information collected in the network. However, those algorithms are centralized.

In [11], the authors use a few nodes equipped with GPS and higher transmission range R called anchors randomly distributed among regular nodes. So two nodes are neighbors if they hear each other and they hear more than T common anchors. A wormhole is detected using two rules: a node should not hear two anchors that are $2R$ apart from each other and should not receive the same message from the same anchor. This algorithm is in fact "half-centralized".

The authors of [6] define the forbidden substructures in the connectivity graphs caused by a wormhole attack. They use a disk graph model to represent a node. First the algorithm determines the packing number $p(S, r)$ (which is the maximum number of nodes in a region S such that each pair of nodes is strictly more than distance r); then it calculates the forbidden parameter f_k which must be more than the packing number for unit distance inside the intersection of two disks of radii k placed at distance 1. The authors develop a cooperative protocol between two nodes u and v: they share their k-hop neighbor lists, then each node tries to compute their common k-hop neighbors $C_k(u, v)$ and to compare its size with f_k. If they do not match, the existence of a wormhole is declared. One limit of the presented approach is the difficulty to compute f_k for realistic models. A centralized approach is also developed where the base station collects information, builds a model of the entire network and tries to detect inconsistencies (potential indicators of wormholes) in this model.

In summary, the methods developed to detect wormhole attacks often use sophisticated hardware (directional antenna, synchronized clocks, GPS,...) that seem not to be a generic solution. Centralized approaches are not really scalable. Finally, the estimation of the forbidden parameter, defined in [6], is harder in random topology. In this case, the parameter is chosen randomly very high and reduced depending on the number of detected false positives.

III. Wormhole detection algorithm

The establishment of a wormhole attack in a sensor wireless networks creates a topology problem and gives a considerable power to the adversary who is able to monitor a large fraction of the network traffic. In this section, we first describe our network model and define the edge-clustering coefficient. Then we present our detection algorithm with no additional hardware component or no timing analysis.

A. Network model and edge-clustering coefficient

1) edge-clustering coefficient: Let us represent a wireless network as a graph $G = (V, E)$, where $V = \{v_1, \cdots, v_n\}$ is a set of *vertices* and $E = \{e_{ij}\}$ is the set of *edges* where

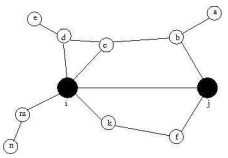


Fig. 2: $CS^{(4)}_{i,j} = 2$: the number of squares is 2 ((i,c,b,j), (i,k,f,j)). $CS^{(4)}_{i,j\setminus b} = Card\{(i,k,f,j)\} = 1$.

e_{ij} connect only the two vertices v_i and v_j. First, we suppose that links between vertices are bidirectional (e.g. $e_{ij} = e_{ji}$) and that the topology is static. We define the neighborhood $V_k(a)$ of a particular vertex $a \in V$ as its k-hop neighbors and the associated degree of the neighborhood as $d_k(a) = |V_k(a)|$ representing the number of nodes in $V_k(a)$ or the number of edges linked to the vertex a.

In [12], the authors define the edge-clustering coefficient (ECC) as the number of triangles to which a given edge belongs, divided by the number of triangles that might potentially include it, given the degrees of the adjacent nodes. More formally, for a edge-connecting the vertices i and j, the edge-clustering coefficient is $C^{(3)}_{i,j} = \frac{z^{(3)}_{i,j}}{\min(d_1(i)-1, d_1(j)-1)}$ where $z^{(3)}_{i,j}$ is the number of triangles built on that edge and $\min(d_1(i)-1, d_1(j)-1)$ is the maximal possible number of them. The authors of [12] generalize this approach considering higher order cycles. In this case, the edge-clustering coefficient of order g for the edge connecting node i to node j, are defined as: $C^{(g)}_{i,j} = \frac{z^{(g)}_{i,j}}{s^{(g)}_{i,j}}$ where $z^{(g)}_{ij}$ is the number of "cyclic structures" of order g the edge (i,j) belongs to, while $s^{(g)}_{ij}$ is the number of all possible cyclic structures of order g that can be built given the degrees of the nodes. The order g represents the number of nodes present on the cyclic structures.

From now, we are going to especially study the coefficient $z^{(g)}_{i,j}$ which will be called $CS^{(g)}_{i,j}$. For sake of simplicity, we also introduce the following coefficient $z^{(g)}_{i,j\setminus X}$ denoted $CS^{(g)}_{i,j\setminus X}$ as the number of cyclic structures that exclude X. An example of such computations for $g = 4$ is given in Figure 2.

2) Wormhole definition: As mentioned before, the placement of wormhole consists in creating a long link between two wormhole nodes. Most of the time, this link is large enough to create shortest paths between two sets of nodes located potentially far away. Thus, from a graph theory point of view, a wormhole link creates false neighborhood information between nodes that believe they are k-hop neighbors when they are not. More precisely, the wormhole link itself supposes that the two wormhole nodes, say X and Y, are 1-hop neighbors when they are not. However, in this case, if two nodes are declared as 1-hop neighbors and if the network is sufficiently dense, those two particular nodes must have some common 1-hop neighbors whereas it is not the case if the corresponding link is a wormhole. More formally, X and Y may be a wormhole link if $((V_1(X)\setminus Y) \cap (V_1(Y)\setminus X) = \emptyset$. This condition could also be expressed using ECC_Num: $CS^{(3)}_{X,Y} = 0$ (i.e. there is no triangle shared between X and Y). Thus, we obtain a necessary condition on the value of $CS^{(3)}_{X,Y}$ to detect a wormhole link (see also Figure 3).

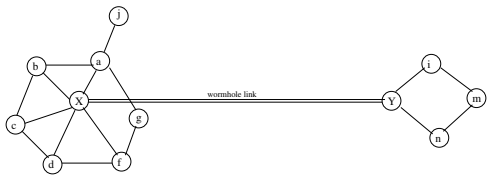


Fig. 3: $X - Y$ is the wormhole link. a computes, for $k \in V_1(X) = \{b,c,d,f,Y\}$, $CS^{(3)}_{a,k\setminus X}$ and $CS^{(4)}_{a,k\setminus X}$: $CS^{(3)}_{a,Y\setminus X} = 0$, $CS^{(4)}_{a,Y\setminus X} = 0$. a declares X as an alert.

Considering the previous simple description of what a wormhole is, how could a 1-hop neighbor a of a wormhole node X detect the wormhole link ? The answer is: a could say that its 1-hop neighbor X is a wormhole node if it could not reach Y, the other wormhole node and a 1-hop neighbor of X, through its "own" 2-hop neighbor-list. In other words and using the CS coefficient, a says that X is a wormhole node if the only triangle that can be constructed between a and Y (considering that a virtual edge exists between a and Y) is (a,X,Y) and if the only squares that can be constructed between a and Y (always considering that a virtual edge exists between a and Y) are $(a,X,\cdot,Y)$ and $(a,\cdot,X,Y)$ where $\cdot$ denotes all the possible nodes. In other words, the number of cyclic structures of order 3 and 4 between a and Y excluding X is null. Thus $CS^{(3)}_{a,Y\setminus X} = 0$ and $CS^{(4)}_{a,Y\setminus X} = 0$. In neighborhood terms, we have Y in $V_1(X)$ such that $Y \notin V_1(a)$ and $Y \notin V_2^{G\setminus X}(a)$ in the subgraph $G \setminus X$. More generally, X is declared as a wormhole by its 1-hop neighbor a if there exists k such as $k \in V_1(X)$ and $k \notin ((V_1(a)) \cup (V_2^{G\setminus X}(a))$, thus $CS^{(3)}_{a,k\setminus X} = 0$ and $CS^{(4)}_{a,k\setminus X} = 0$.

We thus obtain a necessary condition to allow a neighbor of a wormhole node to detect it using neighborhood information. Figure 3 gives an example of this detection supposing that a checks the node X. However and because our condition is only necessary, we sometimes obtain false positives as illustrated on Figure 3: suppose that j wants to test the node a, then it computes $CS^{(3)}_{j,k\setminus a}$ and $CS^{(4)}_{j,k\setminus a}$ for $k \in V_1(a)$ i.e. $k \in \{b,X,g\}$, it obtains for all k, $CS^{(3)}_{j,k\setminus a} = 0$ and $CS^{(4)}_{j,k\setminus a} = 0$ and j declares a as an alert. In this case, we obtain a false positive alert which can sometimes cause disconnections in the network. However, we will see in the next section a simple way to limit the number of false positives.

We could of course generalize our approach saying that if i is a l-hop neighbor of a wormhole node (say X), then it could not reach one of the 1-hop neighbor of X (say Y) through its "own" $l+1$-hop neighbor-list. In this case, we need to compute the corresponding edge-clustering coefficient $CS^{(l+2)}_{i,k\setminus X}$ for all $k \in V_1(X)$. The same kind of generalization could also be considered for a particular wormhole (with more than 2 wormhole nodes) where the distance between the two end-to-end nodes is greater than 1-hop. For sake of simplicity and of efficiency, we have limited our study to the 1-hop neighbors verification, to the case where $g = 3$ or 4.

We have introduced a simple method based upon particular coefficients of the graph theory to detect a wormhole link in a given network. Note that a wormhole link could only be detected if the real distance between the two wormhole nodes is greater than 4-hop due to the computed coefficients. This does not represent a real restriction because the longer the wormhole link is the more efficient it is.

B. Algorithm Description

Our wormhole detection algorithm is to compute particular CS coefficients for each neighbor of a given node. The algorithm is decentralized, distributed and executed locally for each node of the network. Each node searches for particular coefficients in its k-hop neighborhood. The algorithm works for all k values but we explain it only for $k = 2$.

Each node i maintains the list of 1 and 2-hop neighbors. Such information could be easily collected by various upper layer protocols such as routing thus may not present any additional overhead. For example, the neighbor discovering protocol described in [13] could be used: the node i broadcasts a HELLO message containing its identity; every node j that hears the message replays to i including its identity and after, it sends its 1-hop neighborhood $V_1(j)$ and its 2-hop neighborhood $V_2(j)$. At the end of the protocol, each node i has its own $V_1(i)$, $V_2(i)$ and the neighbor lists $V_1(v)$ and $V_2(v)$ of each $v \in V_1(i)$. The algorithm works as follows:

- **Neighborhood discovery.** Each node i determines $V_1(i)$ and $V_2(i)$ and receives $V_1(v)$ and $V_2(v)$ for each $v \in V_1(i)$. It executes the following steps for each j in $V_1(i)$:
- **Compute** CS**.** Node i computes $CS^{(3)}_{i,k\setminus j}$ for every k in $V_1(j)$. If this value is null then i computes $CS^{(4)}_{i,k\setminus j}$. If this new value is also equal to 0, i declares j as a "suspicious node" and puts it in its red list.
- **Isolating phase.** When node i sets the node j in its red list, it broadcasts an alert message containing the j-identity. Each node that hears the message adds j to its red list or increments the corresponding j counter. When a node w gets enough alert messages (this number is denoted by N_{am}), higher than a given threshold T_{am} (for example greater than $d_1(w)/2$), w sends black alert messages to all its direct neighbors to isolate the node j from the network and to limit the damages caused by the wormhole attack.

First, remark that the threshold value for black listing a node influence the performance of the isolating phase: the smaller this value is the more increases the rate of false detection and higher this limit is, more we have chances to miss the real malicious nodes. Note also that each node receiving a black alert accepts it if he has already the concerned node in its red list.

The way the algorithm is presented makes it appear that two nodes make symmetrically the same computation. We can easily use some priority rules based on node IDs to discard this problem.

The algorithm presented above requires only the knowledge of 1-hop and 2-hop neighbor lists for each node. It runs locally and it can be executed periodically or every time the topology has changed in the network but it can be run again only by the nodes affected by this modification. So the wormhole attacks will be detected as soon as they are in place in the network.

The computational complexity of this algorithm is roughly $\mathcal{O}(d^2)$ where d is the average degree of the nodes when a node computes coefficients $CS^{(g)}_{i,k\setminus j}$ for each of its 1-hop neighbors. The approximate message complexity of this algorithm is roughly $\mathcal{O}(d^3)$ because a neighbor of a node i must send its 1-hop and 2-hop neighbor lists. This step could be optimized considering that symmetric computations are performed. The detection algorithm stays however relatively efficient because the average degree d is relatively small in classical networks.

IV. Simulation results

In this section, we present simulation results of our algorithm. We evaluate the probability of wormhole detection, graph disconnection and false positives for various networks. Those simulations are done using the WSnet simulator [14]. First we define the parameters used in our scenarios, then we show the simulation results.

A. Evaluation approach

Our approach is tested on two different node distributions: grid distribution and random distribution. Each simulation is run with about 125 sensor nodes distributed over a square field of 400m by 400m with a single wormhole. The deployed nodes have fixed positions during each simulation. We have used the disk graph connectivity model where each node has a fixed radio range r. Our simulations use the IEEE 802.11 physical and MAC layers which are fully simulated in the WSnet environment. We assume that simulating large sensors networks is not necessary here since our technique is localized.

As noted before, local node density is an important parameter of our model. For this reason, we vary this number in different experiments by changing the number of sensor nodes (between 100 and 150) or by changing the transmission radius of each node. All the values shown in our results are the average of 100 repeated experiments with the same connectivity model using randomly generated topologies or grid topologies. A single wormhole attack is randomly established between two nodes in each experiment with a distance greater than 4-hop. Our algorithm is run with order $g = 3, 4$.

Using this methodology, four probabilities are computed for each set of experiments: the probability of detection, the probability of false positive (assuming that $T_{am} = d_1(w)/2$ for a given node w), the probability of false positive excluding the boundary nodes and the probability of network disconnection (the network is assumed disconnected if any two nodes do not have a path to each other).

As explained in Section III, our algorithm may isolate some legitimate sensor nodes. To reduce this effect, we vary the threshold T_{am} from which the number of alert messages N_{am} trigger the black alert and we simulate its impact on the probability of false positives as shown in Figure 4c.

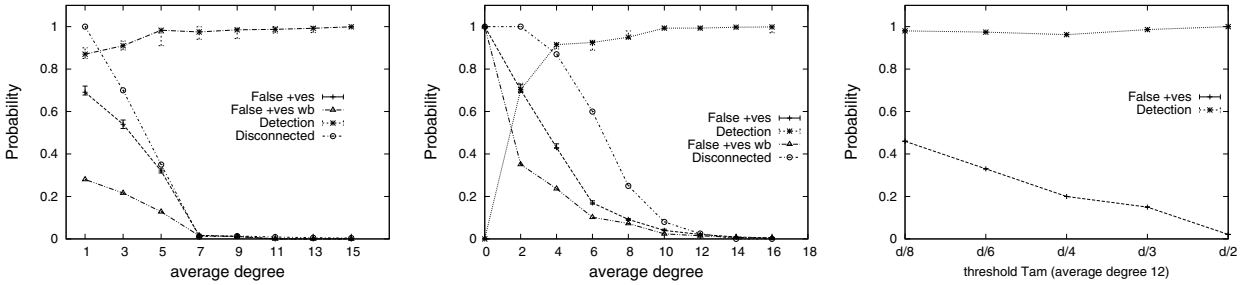


Fig. 4: Probability of wormhole detection, graph disconnection, false positives and false positives excluding boundary nodes: 4a is for grid distribution with unit disk model connectivity; 4b is for random distribution with unit disk model connectivity; 4c represents the impact of the threshold T_{am} on the false positives probability (d represents the local node degree).

B. Results

Figure 4 presents the performance results of our algorithm for grid distributions and random distributions. We can affirm that our approach provides very good results with 100% detection of the wormhole and few false positives for networks having an average degree higher than 7 in all the kind of studied distributions. In summary, the highest detection probability is obtained for topologies where sensor nodes are closer to each other. Clearly this shows the local behaviour of our algorithm. For low density cases, the probability of false positive is relatively high (around 50 % in the worst case) even if about half of the false positives are boundary nodes. The isolation of legitimate nodes declared as wormhole whereas they are not is caused by the lack of information necessary to compute coefficients and is due to the property of non connexity of the subjacent random graph. This could induce disconnections in the network. However, in all cases, the wormhole attack is detected with a high probability which ensure the security of the network. Finally, to reduce the false positive problem, we study the influence of the coefficient T_{am} on the number of false positives generated by our algorithm.

Figure 4c presents the impact of T_{am} on 150 randomly distributed nodes with an average degree equal to 12. We clearly see that the number of false positives decreases when the threshold T_{am} increases. Clearly, the number of nodes that declare legitimated nodes as wormhole is few because the corresponding number of alerts is also few. But the higher the threshold is the lower the number of nodes that really detect a true wormhole node is.

V. Conclusion

In this paper we propose a simple, practical and local algorithm to detect wormholes in wireless networks. Our simulation results show that the probability to detect a single wormhole is really high and that the number of false positives is relatively low as soon as the degree is sufficiently large.

In order to completely isolate a wormhole node, we propose that each node detecting a wormhole node broadcasts the black alert using a secure mechanism as the one proposed in [15] to completely isolate the wormhole node from the rest of the network.

References

[1] C. Karlof and D. Wagner, "Secure routing in wireless sensor networks: attacks and countermeasures," *Ad Hoc Networks*, vol. 1, no. 2-3, pp. 293–315, 2003.

[2] Y.-C. Hu, A. Perrig, and D. B. Johnson, "Packet leashes: A defense against wormhole attacks in wireless networks," in *INFOCOM*, 2003.

[3] D. Johnson, Y. Hu, and D. Maltz, "The dynamic source routing protocol (dsr) for mobile ad hoc networks for ipv4," RFC 4728 - IETF, http://www.rfc-editor.org/rfc/rfc4728.txt, 2007.

[4] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc on-demand distance vector (aodv) routing," RFC 3561 - IETF, http://www.faqs.org/rfcs/rfc3561.html, 2003.

[5] Y.-C. Hu, A. Perrig, and D. B. Johnson, "Ariadne: a secure on-demand routing protocol for ad hoc networks," in *MOBICOM*, I. F. Akyildiz, J. Y.-B. Lin, R. Jain, V. Bharghavan, and A. T. Campbell, Eds. ACM, 2002, pp. 12–23.

[6] R. Maheshwari, J. Gao, and S. R. Das, "Detecting wormhole attacks in wireless networks using connectivity information," in *INFOCOM*. IEEE, 2007, pp. 107–115.

[7] S. Capkun, L. Buttyán, and J.-P. Hubaux, "Sector: secure tracking of node encounters in multi-hop wireless networks," in *SASN*, S. Setia and V. Swarup, Eds. ACM, 2003, pp. 21–32.

[8] L. Hu and D. Evans, "Using directional antennas to prevent wormhole attacks," in *NDSS*. The Internet Society, 2004.

[9] W. Wang and B. K. Bhargava, "Visualization of wormholes in sensor networks," in *Workshop on Wireless Security*, M. Jakobsson and A. Perrig, Eds. ACM, 2004, pp. 51–60.

[10] L. Buttyán, L. Dóra, and I. Vajda, "Statistical wormhole detection in sensor networks," in *ESAS*, ser. Lecture Notes in Computer Science, , vol. 3813. Springer, 2005, pp. 128–141.

[11] R. Poovendran and L. Lazos, "A graph theoretic framework for preventing the wormhole attack in wireless ad hoc networks," *Wireless Networks*, vol. 13, no. 1, pp. 27–59, 2007.

[12] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi, "Defining and identifying communities in networks," *Proceedings of the National Academy of Science of the United States of America, PNAS*, vol. 101, no. 9, pp. 2658–2663, 2004.

[13] K. Heurtefeux and F. Valois, "Self-organisation protocols: Behavior during the sensor network life," in *IEEE International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC)*, Athens, Greece, September 2007.

[14] E. Ben Hamida, G. Chelius and J.-M. Gorce, "Scalability versus Accuracy in Physical Layer Modeling for Wireless Network Simulations," in *22nd ACM/IEEE/SCS Workshop on Principles of Advanced and Distributed Simulation (PADS 2008)*, Rome, Italy, June 2008.

[15] C. Castelluccia, "Securing very dynamic groups and data aggregation in wireless sensor networks," in *IEEE MASS - The Fourth IEEE International Conference on Mobile Ad-hoc and Sensor Systems*, 2007, pp. 1–9.

Energy Friendly Integrity for Network Coding in Wireless Sensor Networks

Anya Apavatjrut, Wassim Znaidi, Antoine Fraboulet, Claire Goursaud, Cédric Lauradoux and Marine Minier
Université de Lyon, INRIA,
INSA-Lyon, CITI laboratory
Email: surname.name@insa-lyon.fr

***Abstract*—The recent advances in information theory and networking have significantly modified the way to disseminate data in wireless sensor networks (WSNs): aggregation, network coding or rateless codes. These new paradigms of dissemination create new threats for security such as pollution attacks. These attacks exploit the difficulty to protect data integrity in those contexts. In this paper, we consider the particular case of xor network coding. We compare the different strategies based on message authentication codes algorithms (MACs) to thwart these attacks. We emphasize the advantages of universal hash functions (UHFs) in terms of flexibility and efficiency. These schemes reduce the energy consumption by 42% and 68% (according to the used protocol) for the relaying nodes over those based on classical cryptographic primitives without any loss in security. The key feature of the UHFs considered here is their homomorphic property ($h(x_1 \oplus x_2) = h(x_1) \oplus h(x_2)$). These homomorphic MACs offer more possibilities for the relying nodes than the classical cryptographic ones: the detection time of a pollution attack can be adjusted to preserve the nodes energy. Moreover, they can be computed with the low resources of a sensor.**

***Index Terms*—Xor network coding, pollution attack, homomorphic MAC, universal hash functions.**

I. Introduction

Securing Wireless sensor networks (WSNs) is a challenging task in hostile environments. The nodes resources are limited in energy and in computational power: the full arsenal of cryptographic primitives and protocols is not adapted to WSNs. The adversary can applied many strategies to corrupt, disrupt the communications and the network: sybil attacks, wormholes,etc. Moreover, the motivations and the goals of the adversary can go beyond the classical properties protected by cryptographic techniques: resources exhaustion, degradation of the QOS (delay),etc.

Parallel to the security advances, new techniques have emerged in information theory to improve the dissemination of information over a network. Many of this new results can be used in WSNs to preserve the nodes energy such as data aggregation (source coding [1], compressed sensing [2]) or network coding [3]. Other algorithms like rateless codes (fountain codes [4]) improve the resiliency of the transmission to packet losses. Many of these transformations have in common that the packets exchanged by the nodes are some linear combinations of the data to be transmitted. The consequence is that it is more difficult for the relaying nodes to know if a data received is legitimate or not. There is an opportunity for an adversary to inject his own data. Such an attack is called a *pollution attack*. A pollution attack can deserve multiples objectives such as data corruption or energy exhaustion. The latter exploits the fact that even with some security measures, illegitimate packets will be carried through the network to be only discarded by the destination. All the energy used to transport the data are lost.

In this work, countermeasures to pollution attacks are investigated on a WSN taking advantage of xor network coding [5]. The core mechanism considered in this paper is message authentication code (MAC) such as HMAC, CBC-MAC and design based on universal hash functions (UHFs). These algorithms aim at protecting both the integrity and the origin of the data. We establish that MACs based on the more classical primitives that are block ciphers or hash functions imply a energetic cost too important for the relying nodes of the network. On the opposite, MACs based on UHFs offer more flexibility in the control of the packet integrity and a lower energy consumption for the relaying nodes. This is a a consequence of the homomorphic properties of the selected UHFs. We argue that low cost homomorphic MACs are critical to conciliate linear network coding and integrity in WSNs.

The contributions of the paper are as follows:

1) Different modes of operations for the relaying nodes in a network using network coding are defined and compared.
2) the performances of the different designs of MACs are compared on MSP430 nodes (energy consumption).
3) the versatility of homomorphic MACs is established.

In Section II, pollution attacks are described as well as related works. The Section III introduces the modes of operations for MAC algorithms in the context of network-coding. The different modes are analyzed in Section IV whereas Section V concludes this paper.

II. Related works on pollution attacks

Pollution attacks first appear in Peer-to-Peer (P2P) applications [6]: malicious users inject unusable or meaningless data in the system to frustrate the users. These attacks reappear with the new networking paradigm that are rateless codes [4], data aggregation or network coding [3]. All this applications share in common the fact that the nodes of the network transmit (linear) combination of the data rather than the raw data. Therefore, protecting the network against pollution attacks

is a challenging task: the nodes need to check that they are dealing with correct data despite the transformations. A overview of the challenges of pollution attacks in network coding applications can be found in [14], [15].

Homomorphic signatures [7], [8], [9], [10] have been proposed to solve the problem of pollution attacks. These techniques are directly inspired by homomorphic encryption schemes. However, they require to compute exponentiation which is far beyond the capacity of a sensor and this even with the use of optimization technique such as batch computation [8].

Agrawal and Boneh have proposed in [11] to use homomorphic MAC to thwart pollution attacks in network coding applications. Their construction is based on an extension of the works of Krawczyk [12] as in this paper. Independently, Znaidi *et al.* in [13] apply the same extension of Krawczyk's work to the problem of data aggregation. From a security point of view, aggregation is very similar to network coding. The scheme of Agrawal and Boneh [11] has been since extended in [16]. However, all these works do not consider the threat of energy exhaustion implied by pollution attacks. We unveil all the possibilities offers by homomorphic MAC to save as much energy as possible.

III. Pollution attack and network coding

To illustrate all the possible strategies for cryptography, we consider the chain topology with network coding [3] in which Eve is a relay for Alice and Bob. Alice sends the message x_1 to Bob, and reciprocally Bob sends x_2 to Alice. At time t_1, Eve has received the messages x_1 and x_2 respectively from Alice and Bob. At time t_2, Eve broadcasts $x_1 \oplus x_2$ to Alice and Bob (Fig. 1). We choose this example rather than the traditional butterfly network because the exposure of the data authentication problem is more easy to follow. It corresponds to a case in WSN in which a sensor Alice sends data to the sink Bob. At the same time, Bob sends a new command to Alice through Eve. Alice, Eve and Bob want to prevent an external adversary to inject corrupted packets.

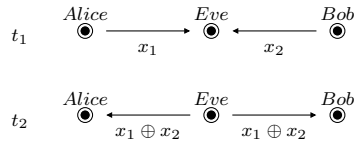


Fig. 1. A cross point in network coding.

The authentication of the messages exchanged in the protocol described in Fig. 1 is now considered. The primary goal of an integrity scheme is to prevent an adversary to tamper with the messages and to forge its own messages. Message authentication codes (MACs), *i.e.* keyed cryptographic hash functions, can be used to thwart those attacks: a tag is appended to the message to check its integrity. Nowadays, three families of MAC algorithms can be found. (1) HMAC [17], and MDx-MAC [18] rely upon cryptographic hash functions for which we are still waiting for the end of the SHA-3 competition. (2) CBC-MAC designs such as [19] depend on the implementation of block ciphers. In this later case, the standard algorithm is the AES. (3) Finally, universal hash functions (UHFs) has been proposed for the design of MAC algorithms. A brief introduction to UHFs is given in Appendix A as well as the design of secure MAC based on UHFs. This class of MAC algorithms offers an unconditional security, and at least the same level of performance than the two previously mentioned designs [20]. This design of MACs based on UHFs is well established in networking community with UMAC [21] and Galois/Counter mode (GCM) [22] of encryption.

A particular class of UHFs is considered here that is ϵ-almost XOR universal (ϵ-AXU) hash functions (see Definition 1 and 2 of Appendix A). These MACs have also an homomorphic property:

$$h_k(x_1 \oplus x_2) = h_k(x_1) \oplus h_k(x_2), \tag{1}$$

with x_1 and x_2 two messages and k a secret authentication key. The design of a MAC from an ϵ-AXU hash function is given by the following equation:

$$h_k(m) = f_k(m) \oplus r, \tag{2}$$

with f_k an universal hash function and r a random pad.

Remark 1: We consider homomorphic MACs with respect to the xor operator (Equation 1). Those fonctions are useful for xor network coding [5], [16] as it will be shown below. Homomorphic universal hash functions have been extended to other finite fields [11], [13], [23] in order to be used for linear network-coding.

Remark 2: Homomorphic properties are well studied for encryption, *i.e.* $E_k(x_1 \oplus x_2) = E_k(x_1) \circ E_k(x_2)$ (see [24] for instance). A well-known example of homomorphic encryption are the stream ciphers. More details, on such encryption schemes and their applications for in-network transformations can be found in [25]. For simplicity, the confidentiality problem is omitted in the remaining parts of the paper.

These three MAC designs are considered to prevent pollution attacks. Four modes of operation to protect the messages integrity are considered. They reflect the operations performed by the relay Eve: Authenticate-Xor-MAC-Forward (AXMF), Authenticate-Xor-Forward (AXF), Xor-Authenticate-Forward (XAF) and Xor-Forward (XF). The last three modes, *i.e.* AXF, XAF and XF, are only available with homomorphic MACs.

A. Authenticate-Xor-MAC-Forward *mode (AXMF)*

In this mode of operation, Alice, Bob and Eve share a secret key k. Note that we do not discuss in this paper the key distribution (see [26] for instance). The Authenticate-Xor-MAC scheme can be implemented with HMAC, MDxMAC or CBC-MAC. We particularly consider the implementation of this strategy with HMAC-SHA-1 [17] used in IPV6 and AES-128-CBC-MAC. The communication consists in three steps:

1) **Data generation**. Alice computes $d_1 = h_k(x_1)$ the digest of her message, and reciprocally, Bob computes

$d_2 = h_k(x_2)$. Then, the messages x_1, d_1 and x_2, d_2 are sent to Eve.

2) **Authenticate-Xor-MAC-Forward**. At time t_1, Eve has received x'_1, d'_1 and x'_2, d'_2. Eve forwards $x'_1 \oplus x'_2$ if and only if $h_k(x'_1) \stackrel{?}{=} d'_1$ and $h_k(x'_2) \stackrel{?}{=} d'_2$ (AUTHENTICATE). As we assume that the MAC algorithm is secured, the success of the verify step implies that $x'_1 = x_1$ and $x'_2 = x_2$. If these messages are successfully authenticated, she computes $m = x_1 \oplus x_2$ (XOR) and the digest of this value $d_3 = h_k(m)$ (MAC). Eve eventually broadcasts $m, h_k(m)$ to Alice and Bob (FORWARD).
3) **Delivery and verification.** Alice and Bob receive respectively m_a, d_a and m_b, d_b. They verify their respective values $h_k(m_a) \stackrel{?}{=} d_a$ and $h_k(m_b) \stackrel{?}{=} d'_b$. When the corresponding verification succeeds, Alice obtains $x_2 = m_a \oplus x_1$ or/and Bob obtains $x_1 = m_b \oplus x_2$.

The AUTHENTICATE-XOR-MAC-FORWARD mode is described in Fig. 2.

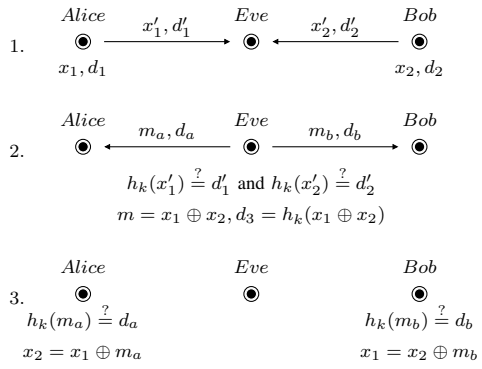


Fig. 2. AUTHENTICATE-XOR-MAC-FORWARD.

Remark 3: The scheme described previously is secure if a single exchange needs to be supported, *i.e.* replay attacks are not considered here. Replay can be easily defeated by adding counters to the message. They are inserted in the computation of CBC-MAC as the public initial value (IV). They are skipped here for clarity.

B. AUTHENTICATE-XOR-FORWARD *mode (AXF)*

Alice, Bob and Eve share a secret key k. The use of ϵ-AXU UHFs is assumed for the MAC and the random numbers required for the MAC are produced by the AES in counter mode (AES-CTR). The Toeplitz hashing proposed by Krawczyk [12] (see Section A) is used in this paper for the function h. The AUTHENTICATE-XOR-FORWARD takes full advantage of the homomorphic property (Equation 1) of the underlying MAC. Eve does not evaluate explicitly a MAC but only "xor" the tags of different messages. The mode is working as follows:

1) **Data generation**. This step is exactly the same as in the AUTHENTICATE-XOR-FORWARD mode.
2) **Authenticate-Xor-Forward**. At time t_1, Eve has received x'_1, d'_1 and x'_2, d'_2. Eve forwards the messages if and only if $h_k(x'_1) \stackrel{?}{=} d'_1$ and $h_k(x'_2) \stackrel{?}{=} d'_2$ (AUTHENTICATE). She computes $m = x'_1 \oplus x'_2$ (XOR) and $d_3 = d'_1 \oplus d'_2$. Then, she computes $h_k(m)$ by herself and she verifies the homomorphic property (Equation 1): $h_k(m) = d'_1 \oplus d'_2$. When the verification is successful, Eve broadcasts eventually m, d_3 to Alice and Bob (FORWARD).
3) **Delivery and verification.** This step is exactly the same as in the AUTHENTICATE-XOR-FORWARD mode.

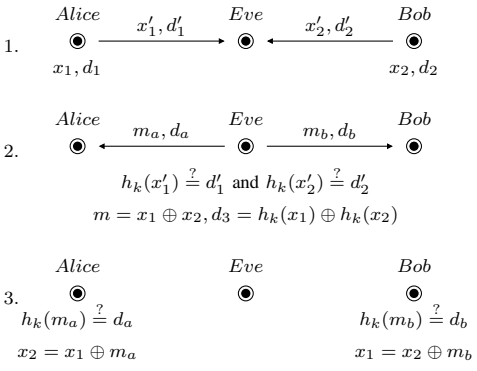


Fig. 3. AUTHENTICATE-XOR-FORWARD.

AUTHENTICATE-XOR-FORWARD mode of operation is depicted in Fig. 3.

C. XOR-AUTHENTICATE-FORWARD *mode (XAM)*

AUTHENTICATE-XOR-FORWARD mode of operation performs the verification prior to the data combination. It is also possible to postpone the verification after the combination of the data. This is possible because the combination operation is neutral for the data integrity. The idea of the XOR-AUTHENTICATE-FORWARD mode (Fig. 4) is to verify the correctness of the data by verifying the homomorphic property. The mode proceeds as follows:

1) **Data generation**. This step is exactly the same as AUTHENTICATE-XOR-MAC-FORWARD. The messages $x_1, d_1 = h_k(x_1)$ and $x_2, d_2 = h_k(x_2)$ are sent to Eve.
2) **Xor-Authenticate-Forward**. At time t_1, Eve has received x'_1, d'_1 and x'_2, d'_2. Eve forwards the messages if and only if $h_k(x'_1) \stackrel{?}{=} d'_1$ and $h_k(x'_2) \stackrel{?}{=} d'_2$ (AUTHENTICATE). If these messages are authenticated, she computes $m = x'_1 \oplus x'_2 = x_1 \oplus x_2$ (XOR) and she exploits Equation 1 to compute the MAC of m: $h_k(x'_1 \oplus x'_2) = d'_1 \oplus d'_2$. Then, Eve broadcasts eventually $m, h_k(m)$ to Alice and Bob (FORWARD).
3) **Delivery and verification.** Alice and Bob receive respectively m_a, d_a and m_b, d_b. Alice computes $x''_2 = m_a \oplus x_1$, $d''_2 = h_k(m_a) \oplus d_1$ and verifies that $h_k(x''_2) \stackrel{?}{=}$

d''_2. The same computation and verification are performed by Bob: $x''_1 = m_b \oplus x_2$, $d''_1 = m_b \oplus d_2$ and verifies that $h_k(x''_1) \stackrel{?}{=} d''_1$.

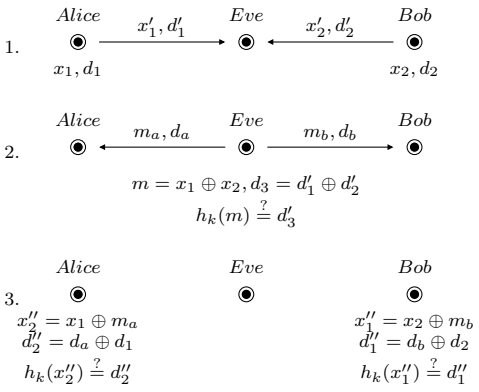


Fig. 4. XOR-AUTHENTICATE-FORWARD.

D. XOR-FORWARD *mode (XF)*

The XOR-FORWARD mode postpones any verification to the legitimate receiver of the message: Eve does not authenticate the messages she receives. The generation step and the delivery/verification step are unchanged compared to the XAF mode. When Eve has received x'_1, d'_1 and x'_2, d'_2, she broadcasts $x'_1 \oplus x'_2$, $d'_1 \oplus d_2$ to Alice and Bob without further verification. The XOR-FORWARD has two advantages for the relaying node: Eve does not need to know the secret key k which will eventually simplify the key distribution and she does not do any verification.

The drawback is that the polluted packets are still forwarded. This can waste the energy and the time of all the nodes. The XF and XAF mode could be combined together in the network to find a n appropriate trade-off between the energetic cost when no attack occur and the energetic cost when an adversary inject corrupted packets. To optimize the speed of detection of a pollution attack the AXF and XAF are to be considered.

IV. COMPARISON AND ANALYSIS

In this section, we consider the performance of the different modes of protection against pollution attack.

A. Simulation setup

Performance results have been obtained using WSim and WSNet open source simulation tools [27]. Performance estimations have been performed on a MSP430 based platform similar to TelosB nodes (Senslab v1.4). These nodes use a Ti MSP430f1611 micro-controller and a Ti CC2420, 802.15.4 compliant, radio device among other devices.

WSim[1] is a full system emulator that takes as input the full ELF firmware generated by cross compilation tools. The toolchain used in the experiment is GCC 4.4.3 port for MSP430[2]. Execution of the embedded software is cycle accurate, time evaluation is also accurate thanks to the full emulation of the micro-controller clocking system. When coupled with WSNet[3], an event driven network and physical layer simulator, WSim can estimate the correctness of applications and drivers for network devices at a byte precise level. These two simulation tools have been used to benchmark and generate non intrusive execution traces of all the MAC algorithms.

The eSimu energy estimation tool [27] can then estimate the energy profile of the execution trace. eSimu uses micro benchmarks to build platform energy profiles. eSimu achieves an error rate less than 5% for energy estimation and annotation at source level (C code) when used for full applications including complex devices such as radio transmitter or flash memories. The error rate drops to less than 1% for computation intensive algorithm for which only the energy spent in the microcontroller is computed.

B. Comparison of MACs

As a preliminary, we provide the performance of the MAC algorithms. The data to be authenticated are 20 bytes long. The keys used for all the algorithms are of size 128 bits. HMAC provides MAC of 160 bits while AES-CMAC and Toeplitz hashing provide MAC of 128 bits due to their inherent designs. The main characteristics of the computation of the different MACs are summarized in Table I. The HMAC algorithm consists in two calls to a cryptographic hash function (SHA-1). For CBC-MAC, two calls to a block cipher (AES-128) are done. For the Toeplitz hash, we need to compute the universal hash function h_k and to xor the result with a nonce produced by the AES-128 in the CTR mode (one call to AES-128).

		Calls to	
MAC	Output size (bits)	AES-128	SHA-1
HMAC	160	0	2
CBC-MAC	128	2	0
Toeplitz	128	1	0

TABLE I
CHARACTERISTICS OF THE MACS FOR 20 BYTES PLAINTEXTS.

The processing time and the energy consumption of HMAC, AES-CBC and Toeplitz are given in Table II along with the performance of AES-128 and SHA-1. We also benchmark a 512-bit exponentiation to give an order of the cost of homomorphic signatures based on exponentiation. This operation is performed in 15052 ms and it consumes 1000 times the energy of HMAC.

The Toeplitz hashing reduces the energy consumption by 65% over HMAC and 23% over AES-CBC.

[1] http://wsim.gforge.inria.fr/

[2] http://mspgcc4.sf.net/

[3] http://wsnet.gforge.inria.fr/

Algorithm	Energy in 10^{-4} $Joules$	Delay in ms
SHA-1	0.623	4.64
AES-128	0.473	3.53
HMAC	2	14.84
CBC-MAC	1.476	11.045
Toeplitz	1.201	8.976

TABLE II
EVALUATION OF MACS ON TI MSP430 AT 8MHZ.

C. Simulation results

A detailed analysis of the energy consumption of the nodes is given in the Fig. 5 (a)–(e). The case of a communication without attack is first considered. We have decomposed the energy consumed by a node into four part:

1) Reception cost: all the radio operations performed by a node to receive a packet.
2) Verification cost: all the computation to authenticate a packet.
3) Transformation cost: it includes the combination of the packets and the generation of the tags associated to them.
4) Emission cost: all the radio operations performed by a node to emit a packet.

The AXMF mode with either HMAC (Fig 5.(a)) or CBC-MAC (Fig 5.(b)) are the most restricting for the relaying node. Eve consumes with HMAC more energy to relay (8.29×10^{-4} J) than Alice or Bob to generate and emit the data (5.53×10^{-4} J). The situation is almost the same with CBC-MAC. The situation is more fair with AXF (Fig 5.(c)): 5.51×10^{-4} J are consumed by Eve and 4.68×10^{-4} J for Alice or Bob. The XAF (Fig 5.(d)) and XF (Fig 5.(e)) provides respectively and improvement of 18% and 55% over AXF.

The reception and emission cost are almost the same for all the modes. They become very important when we distinguish the case of communication with or without attack. Indeed, Eve will relay information depending on the success of her verification. This operation implies at least an emission for Eve and two receptions, one for Alice and one for Bob. The energy consumption of all the nodes of the network is given in Fig. 6 for the AXF, XAF and XF. The reader must bear in mind that the energy consumption when the communication is not attacked correspond to a case in which Alice and Bob obtain data at the end. The energy consumed by the nodes ($E_{noattack}$) produces the expected results. When the network is under attack, Alice and Bob are not going to achieve the expected results all the energy (E_{attack}) is wasted. XF is the most interesting when no attack occurs while XAF performs better under an attack.

The choice of a mode of operation depends on the probability p of pollution attack to occur. From the energetic cost of a mode $\mathcal{M}$, a probable energy function $S(\mathcal{M})$ can be defined:

$$S(\mathcal{M}) = p \cdot E_{attack}(\mathcal{M}) + (1-p) \cdot E_{noattack}(\mathcal{M}),$$

where $E_{attack}(\mathcal{M})$ and $E_{noattack}(\mathcal{M})$ denote respectively the energy spend by all the nodes of the network with the mode $\mathcal{M}$

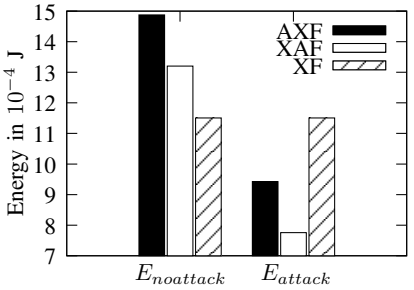


Fig. 6. Energy consumption of the network with/without an attack.

during or not an attack. For the network of Fig. 1, the probable energy spent for AXF, XAF and XF is given in Fig. 7.

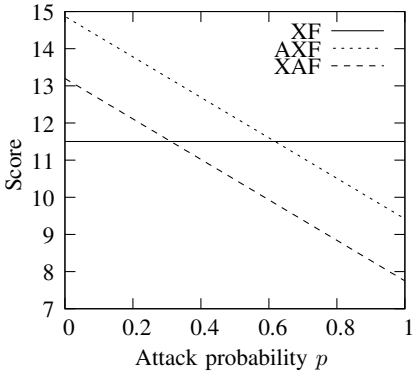


Fig. 7. Probable energy consumed by the overall network for the different modes depending on the attack probability.

The best mode of operation for preserving the energy is XOR-FORWARD if $p < 0.31$. Otherwise, XOR-AUTHENTICATE-FORWARD should be used.

V. CONCLUSION AND OPEN PROBLEMS

We have studied the efficiency of MACs on sensors in terms of delay and energy. From this study, we have capitalize to propose and study the natural modes of operation of MACs. We show that in an energy efficient scheme fighting pollution attacks, a good design consists in a combination of relays using AUTHENTICATE-XOR or XOR modes. An ongoing work will consist to consider larger networks with more complex combination of the data. An optimization process will be used to attribute which nodes use AUTHENTICATE-XOR or XOR (or their equivalent).

Homomorphic MACs applies well in our context because of the linearity of the considered transformations. However, some network-coding problems require to use non-linear methods [28]. For this class of problems, the only solution available against pollution attacks is AUTHENTICATE-XOR-MAC. Adapted integrity schemes are to be unveiled in this case.

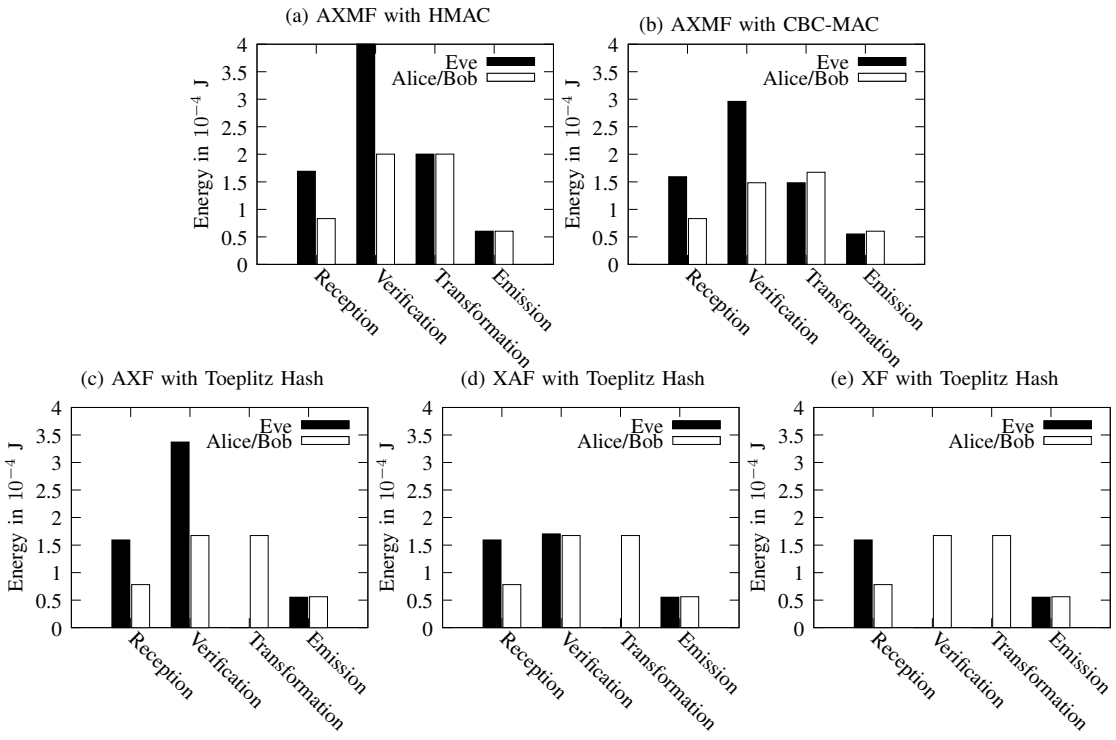


Fig. 5. Energy cost for the different modes of operation.

An important part of the existing works on in-network transformations for WSNs is dedicated to data aggregation. If many solutions exists ([29], [30], [31], [13]), they essentially concern linear aggregations, *e.g.* the aggregation function is the mean. The recent advances in source coding or compressed sensing have introduced new non-linear aggregation functions that must also be considered. If the confidentiality of source coding has been studied, integrity protection for those schemes is still an open problem.

REFERENCES

[1] S. S. Pradhan and K. Ramchandran, "Distributed source coding using syndromes (DISCUS): design and construction," *IEEE Transactions on Information Theory*, vol. 49, no. 3, pp. 626–643, 2003.

[2] D. Baron, M. F. Duarte, M. B. Wakin, S. Sarvotham, and R. G. Baraniuk, "Distributed Compressive Sensing," *CoRR*, vol. abs/0901.3403, 2009.

[3] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, 2000.

[4] M. Luby, "Lt codes," in *Symposium on Foundations of Computer Science - FOCS 2002*. IEEE Computer Society, 2002, pp. 271–.

[5] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft, "XORs in the air: practical wireless network coding," *IEEE/ACM Transactions on Networking*, vol. 16, no. 3, pp. 497–510, 2008.

[6] J. Liang, R. Kumar, Y. Xi, and K. W. Ross, "Pollution in p2p file sharing systems," in *IEEE INFOCOM 2005*. IEEE, 2005, pp. 1174–1185.

[7] M. N. Krohn, M. J. Freedman, and D. Mazières, "On-the-Fly Verification of Rateless Erasure Codes for Efficient Content Distribution," in *IEEE Symposium on Security and Privacy - S&P 2004*. IEEE Computer Society, 2004, pp. 226–240.

[8] C. Gkantsidis and P. Rodriguez, "Cooperative Security for Network Coding File Distribution," in *IEEE INFOCOM 2006*. IEEE, 2006.

[9] D. Charles, K. Jain, and K. Lauter, "Signatures for network coding," *International Journal in Information and Coding Theory*, vol. 1, no. 1, pp. 3–14, 2009.

[10] D. Boneh, D. Freeman, J. Katz, and B. Waters, "Signing a linear subspace: Signature schemes for network coding," in *Public Key Cryptography - PKC 2009*, ser. Lecture Notes in Computer Science 5443. Springer Verlag, 2009, pp. 68–87.

[11] S. Agrawal and D. Boneh, "Homomorphic MACs: MAC-Based Integrity for Network Coding," in *Applied Cryptography and Network Security - ACNS 2009*, ser. Lecture Notes in Computer Science 5536. Springer Verlag, 2009, pp. 292–305.

[12] H. Krawczyk, "LFSR-based Hashing and Authentication," in *Advances in Cryptology - CRYPTO '94*, ser. Lecture Notes in Computer Science 839. Springer-Verlag, 1994, pp. 129–139.

[13] W. Znaidi, C. Lauradoux, and M. Minier, "Aggregated Authentification (AMAC) using Universal Hash Functions," in *International ICST Conference on Security and Privacy in Communication Networks - SecureComm 2009*, ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 17. Springer Verlag, 2009, pp. 248–264.

[14] J. Dong, R. Curtmola, and C. Nita-Rotaru, "Practical defenses against pollution attacks in intra-flow network coding for wireless mesh networks," in *ACM conference on Wireless network security - WiSec '09*. ACM, 2009, pp. 111–122.

[15] ——, "Secure network coding for wireless mesh networks: Threats, challenges, and directions," *Computer Communication*, vol. 32, no. 17, pp. 1790–1801, 2009.

[16] Z. Yu, Y. Wei, B. Ramkumar, and Y. Guan, "An efficient scheme for securing xor network coding against pollution attacks," in *IEEE INFOCOM 2009*, 2009, pp. 406–414.

[17] H. Krawczyk, M. Bellare, and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication," 1997, rFC 2104.

[18] B. Preneel and P. C. van Oorschot, "MDx-MAC and Building Fast MACs from Hash Functions," in *Advances in Cryptology - CRYPTO '95*, ser. Lecture Notes in Computer Science 963. Springer Verlag, 1995, pp. 1–14.
[19] J. Black and P. Rogaway, "CBC MACs for Arbitrary-Length Messages: The Three-Key Constructions," *Journal of Cryptology*, vol. 18, no. 2, pp. 111–131, 2005.
[20] W. Nevelsteen and B. Preneel, "Software performance of universal hash functions," in *Advances in Cryptology - EUROCRYPT '99*, ser. Lecture Notes in Computer Science 1592. Springer Verlag, 1999, pp. 24–41.
[21] J. Black, S. Halevi, H. Krawczyk, T. Krovetz, and P. Rogaway, "UMAC: Fast and Secure Message Authentication," in *Advances in Cryptology - CRYPTO '99*, ser. Lecture Notes in Computer Science 1666. Springer Verlag, 1999, pp. 216–233.
[22] D. A. McGrew and J. Viega, "The Security and Performance of the Galois/Counter Mode (GCM) of Operation," in *Progress in Cryptology - INDOCRYPT 2004*, ser. Lecture Notes in Computer Science 3348. Springer Verlag, 2004, pp. 343–355.
[23] P. Sarkar, "A New Universal Hash Function and Other Cryptographic Algorithms Suitable for Resource Constrained Devices," Cryptology ePrint Archive, Report 2008/216, 2008, http://eprint.iacr.org/.
[24] C. Fontaine and F. Galand, "A survey of homomorphic encryption for non-specialists," *EURASIP Journal on Information Security*, p. Article ID 13801, 2007.
[25] C. Castelluccia, A. C.-F. Chan, E. Mykletun, and G. Tsudik, "Efficient and provably secure aggregation of encrypted data in wireless sensor networks," *ACM Transactions on Sensor Networks*, vol. 5, no. 3, pp. 1–36, 2009.
[26] Y. Xiao, V. K. Rayi, B. Sun, X. Du, F. Hu, and M. Galloway, "A survey of key management schemes in wireless sensor networks," *Computer Communications*, vol. 30, no. 11-12, pp. 2314–2341, 2007.
[27] A. Fraboulet, G. Chelius, and E. Fleury, "Worldsens: development and prototyping tools for application specific wireless sensors networks," in *6th ACM/IEEE international conference on Information Processing in Sensor Networks (IPSN 2007)*. ACM, 2007, pp. 176–185.
[28] S. Riis, "Linear versus non-linear boolean functions in network flow," in *Conference on Information Sciences and Systems - CISS 2004*, 2004.
[29] B. Przydatek, D. Song, and A. Perrig, "SIA: Secure Information Aggregation in Sensor Networks," in *ACM International conference on Embedded networked sensor systems - SenSys 2003*, 2003.
[30] L. Hu and D. Evans, "Secure aggregation for wireless networks," in *In Workshop on Security and Assurance in Ad hoc Networks*, 2003, pp. 384–394.
[31] H. Chan, A. Perrig, and D. Song, "Secure hierarchical in-network aggregation in sensor networks," in *ACM conference on Computer and communications security - CCS '06*. ACM, 2006, pp. 278–287.
[32] L. Carter and M. N. Wegman, "Universal Classes of Hash Functions (Extended Abstract)," in *ACM Symposium on Theory of Computing - STOC '77*. ACM, 1977, pp. 106–112.
[33] ——, "Universal Classes of Hash Functions," *Journal of Computer and System Sciences - JCSS*, vol. 18, no. 2, pp. 143–154, 1979.
[34] V. Shoup, "On Fast and Provably Secure Message Authentication Based on Universal Hashing," in *Advances in Cryptology - CRYPTO '96*, ser. Lecture Notes in Computer Science 1109. Springer-Verlag, 1996, pp. 313–328.
[35] H. Handschuh and B. Preneel, "Key-Recovery Attacks on Universal Hash Function Based MAC Algorithms," in *Advances in Cryptology - CRYPTO 2008*, ser. Lecture Notes in Computer Science 5157. Springer Verlag, 2008, pp. 144–161.

APPENDIX

A universal hash function is a family of functions indexed by a parameter called the key and it must verify that the probability over all keys that all distinct inputs collide is small. This notion was introduced by Carter and Wegman in [32]. A relaxed notion called ϵ-almost universal was also introduced in [33]:

Definition 1: Let f_k be a function of an (ℓ, n)-family H from an ℓ-bit set to an n-bit set with the parameter k taken in a set $\mathcal{K}$. The family H is ϵ-almost universal if the probability of collisions for a random distribution of the value k over the set $\mathcal{K}$ (i.e. $Pr_k(f_k(M) = f_k(M')), \forall k \in_R \mathcal{K}$) is smaller than ϵ.

Definition 2: We also say that a family of functions H is $\oplus$-linear if for all M, M', we have $f_k(M \oplus M') = f_k(M) \oplus f_k(M')$ for all instance f_k in H.
A family H of functions that is at the same time ϵ-almost universal and $\oplus$-linear is said to be ϵ-almost XOR universal (ϵ-AXU). In this case, it must verify that the associated differential probability for a random distribution of the value k over the set $\mathcal{K}$ is bounded by ϵ, i.e. $\forall(M, M', a), Pr_k(f_k(M) - f_k(M') = a) \leq \epsilon$.

The universal hash functions can be used for message authentication if the output is processed with another function. A MAC design using such a family of functions assumed the following scenario: the parties have already exchanged their secret key k, then to exchange a message M of length ℓ, the sender sends M and the corresponding *tag*:

$$h_k(M) = f_k(M) \oplus r. \tag{3}$$

The shared secret key k is thus composed of a particular f_k function drawn randomly from an (ℓ, n)-family of hash functions and a random pad r. At reception, the receiver verifies the tag $h_k(M)$, corresponding with the MAC that will be recomputed and checked for consistency. In practice, the fingerprint $f_k(M)$ will be encrypted with a stream cipher that will produce r.

Krawczyk has shown in [12] that the design of MAC of the above kind (i.e. combined with a one-time pad) requires only to have a family of functions that is ϵ-almost universal. Moreover, the family of functions can also be $\oplus$-linear. Thus, we can used for message authentication family of functions that are ϵ-almost XOR universal leading to homomorphic MAC constructions.

Many universal, ϵ-almost universal and ϵ-AXU hash families have been proposed in the literature to build MACs (see for example [34], [23], [35]). We mainly focus here on one proposal done by H. Krawczyk in 1994 in [12] that is really suitable for constraint environments.

A. Toeplitz hashing

In the same article, Krawczyk introduced a second construction based upon random matrices. More precisely, given A a boolean Toeplitz matrix of size $n \times \ell$ (i.e. each lower diagonal is fixed, i.e. if $k - i = l - j$ for all indices then $A_{i,j} = A_{k,l}$) and given a message M of size ℓ, the universal hash function $h_A(M)$ is the binary multiplication of the matrix A by the column vector composed of message bits of M: $h_A(M) = A \cdot M$.

A simple and practical method to build such matrices is the LFSR use: from a particular irreducible polynomial of degree n over $\mathbb{F}_2$ $q(x)$, we build the corresponding LFSR of size n bits. The output sequence is denoted $s_0, s_1, \cdots$. The initial state of the LFSR $s = (s_0, s_1, \cdots, s_{n-1})$ will represent a part of the secret key. More precisely, for each irreducible polynomial $q(x)$ and for each non-zero initial state of the

LFSR, we associate the hash function $h_{q,s}(M)$ defined as the linear combination $\bigoplus_{j=0}^{\ell-1} M_j \cdot (s_j, s_{j+1}, \cdots, s_{j+n-1})$ where M_j is the bit number j of M. In other words, at each clock, the LFSR updates its internal state taking into account each message bit. This hash functions family is $\oplus$-linear, ϵ-almost universal (with $\epsilon \leq \frac{n+\ell}{2^{n-1}}$) and ϵ-almost XOR universal (with $\epsilon \leq \frac{\ell}{2^{n-1}}$).