Protocoles cryptographiques pour l'authentification numérique et le respect de la vie privée

Quentin Alamélou

UNIVERSITÉ DE LIMOGES
ÉCOLE DOCTORALE SCIENCES ET INGÉNIERIE POUR L'INFORMATION
FACULTÉ DES SCIENCES ET TECHNIQUES

Département Mathématiques & Sécurité de l'Information
Laboratoire XLIM-UMR CNRS n°7252

Thèse

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE LIMOGES

Discipline : Informatique et Applications

présentée et soutenue par

Quentin ALAMÉLOU

le 09 Mai 2017

Protocoles cryptographiques pour l'authentification numérique et le respect de la vie privée.

Thèse dirigée par Philippe Gaborit et encadrée par Stéphane Cauchie.

JURY :

Philippe GABORIT	Professeur, Université de Limoges	Directeur
Ayoub OTMANI	Professeur, Université de Rouen	Rapporteur
Gilles ZÉMOR	Professeur, Université de Bordeaux	Rapporteur
Olivier BLAZY	Maître de conférence, Université de Limoges	Examinateur
Stéphane CAUCHIE	Ingénieur Docteur, equensWorldline, Seclin	Examinateur et Encadrant industriel

"Je voulais être seul mais trop tard j'étais déjà né."

Booba.

Contents

List of Figures

List of Tables

Introduction générale

Contexte de la Thèse

Les travaux de thèse présentés dans ce manuscrit ont été réalisés dans le cadre d'un dispositif CIFRE -Conventions Industrielles de Formation par la REcherche- établi entre Worldline[1,2] et le laboratoire XLIM-DMI de l'Université de Limoges. Même si l'objectif n'est pas ici de présenter un organigramme de equensWorldline, nous proposons ces quelques lignes pour mieux situer le contexte dans lequel ces travaux ont été entrepris. Devenue filiale autonome, depuis 2014, de la SSII Atos, equensWorldline est un acteur majeur des paiements de proximité et le leader mondial dans le domaine des paiements sécurisés en ligne.

Parmi les nombreuses solutions de paiement que fournit le groupe, le service ITA –pour *Identity, Trust and Authentication*– s'est notamment concentré sur la solution d'authentification *Trusted Authentication* (TA). L'objectif initial était de proposer une solution *d'authentification forte* d'un utilisateur via son smartphone sans mise en jeu d'élément sécurisé (*e.g.* une carte SIM). À long terme, l'idée sera d'enrichir cette solution pour authentifier un utilisateur, à partir du contexte, de ses différentes terminaux (téléphone de type *smartphone*, tablette, ordinateur, montre connectée, ...), de ses préférences, de son environnement (chez lui, au travail, avec des amis, ...) voire de son réseau social[3]. Les travaux présentés dans ce manuscrit ont pour but d'accompagner la transition entre l'objectif initial –et toujours d'actualité– et le bouquet grandissant de services tout en s'assurant que les contraintes industrielles d'un point de vue du respect de la vie privée des utilisateurs finaux restent respectées. Nous proposons une brève chronologie des différents stades du projet TA, à laquelle nous nous référerons tout au long de ce manuscrit, pour faire le lien entre l'avancement du projet TA et l'avancement des travaux de thèse décrits ici.

- **TA1.** Pour répondre à un besoin client, une solution d'authentification purement logicielle a été proposée et un brevet déposé [2]. Industrialisé depuis 2012, le stade TA1 fournit une solution d'authentification forte à de nombreux clients qui sont majoritairement des banques parmi lesquelles la Société Générale (SG), Bancontact (BC), La Banque Postale, la Banque Populaire, BNP Paribas. Cette description du contexte et de la solution au commencement de la thèse fera l'objet du chapitre 2 ;

- **TA2.** Cette étape correspond au démarrage de la thèse en Janvier 2014. Conformément à TA1, la solution d'authentification est industrialisée mais présente certaines limitations. Ces limitations concerneront, entre autres, le manque d'expressivité de la solution TA1, la génération/gestion de clé et la gestion de

[1] `http://worldline.com/fr/accueil.html`.
[2] equensWorldline depuis 2016.
[3] Présentation commerciale `https://www.youtube.com/watch?v=VSmNVtdRwQ0`.

données identifiantes bruitées dans le respect de la vie privée. Nous apporterons des solutions aux limitations du stade TA1 en couplant le domaine du *fingerprinting* à celui des Fuzzy Extractors (Dodis *et al*, Eurocrypt 2004). Un Fuzzy Extractor peut se voir comme un générateur de clés cryptographiques dont l'entrée pourra légèrement changer –relativement à une certaine métrique– sans que la valeur de la clé extraite ne soit impactée. Ces travaux, décrits dans le chapitre 3, ont fait l'objet d'un dépôt de brevet [3] et d'une participation à l'Atelier sur la Protection de la Vie Privée (APVP 2014) [4]. La fin de ce chapitre résumera les limitations théoriques et pratiques à un déploiement industriel du stade TA-2. En particulier, nous serons confrontés à la problématique de réutilisation des Fuzzy Extractors identifiée par Boyen (CCS 2004). D'autre part, en proposant le domaine du *fingerprinting* comme nouveau cas d'usage pour les Fuzzy Extractors, nous serons amenés à gérer des données subissant de fortes variations au cours du temps ;

- **TA3.** L'accession au stade TA3, décrite au Chapitre 4, se fera par la résolution des limitations théoriques du stade TA2 parmi lesquelles figure la réutilisabilité[4] des Fuzzy Extractors (Boyen, CCS 2004). Pour ce faire, nous y proposerons un framework générique permettant de convertir tout Fuzzy Extractor classique en un Fuzzy Extractor réutilisable. Ce framework sera instancié dans le cas de la différence d'ensembles qui est l'une, si ce n'est la métrique la plus adaptée aux domaines de la biométrie et du *fingerprinting*. D'autre part, cette application des Fuzzy Extractors au domaine du *fingerprinting* nous a conduit à considérer des Fuzzy Extractors Adaptatifs (*Adaptive Fuzzy Extractors*) pour répondre à la durée de vie réduite des valeurs considérées. Ces travaux sont en cours de soumission et disponibles sur ePrint [5] ;

- **TA3-Ind.** Bien que l'objet théorique présenté au Chapitre 4 permettra une accession au stade TA-3, certaines limitations pratiques –sur l'entropie portée par les nouvelles données d'authentification que nous allons considérer– ne seront pas résolues. Des études, à grand échelle, vont être lancées en 2017 afin d'y répondre et permettre une industrialisation du stade TA3.

- **TA4.** Prenant en compte la croissance de l'Internet des Objets, un objectif à plus long terme de la solution TA est d'authentifier un utilisateur à partir du contexte, de son environnement (chez lui, au travail, avec des amis, ...) et de ses différentes terminaux connectés (smartphone, tablette, ordinateur, montre connectée, ...). Toujours dans l'optique d'assurer la vie privée, un utilisateur devrait alors pouvoir

[4]En fait, Boyen a exhibé qu'utiliser la primitive d'enrôlement d'un Fuzzy Extractor un trop grand nombre de fois pourrait pourrait mettre en danger le secret utilisateur censé être protégé.

s'authentifier sans qu'un serveur ne puisse savoir quels appareils et/ou facteurs déterminants ont été mis en jeu. Nous nous sommes alors intéressés au concept de signature de groupe pour répondre à cette problématique. Ces travaux, décrits aux Chapitres 5 et 6 ont conduit à la conception de deux schémas de signature de groupe résistants à l'ordinateur quantique, respectivement basés sur les métriques de Hamming et Rang. Ces schémas ont été publiés dans les conférences *The Ninth International Workshop on Coding and Cryptography 2015* (WCC 2015) et *International Workshop on the Arithmetic of Finite Fields* (WAIFI 2016) [6, 7] ainsi que dans le journal *Design, Codes and Cryptography*(DCC) [8].

Contributions industrielles et académiques

Les contributions de cette thèse sont à lier au jalons présentés ci-dessus. Nous les présentons en respectant cette description :

1. En appliquant les Fuzzy Extractors aux données numériques de nos utilisateurs, nous avons amélioré la solution TA tant au niveau de son expressivité que du respect de la vie privée. Le module FAST, pour *Fuzzy Authentication for Security and Trust* décrit au Chapitre 3, englobant ce nouveau cas d'usage et dédié à la génération de clés à partir de données utilisateur potentiellement bruitées, a conduit à un dépôt de brevet [3]. L'industrialisation de ce module a commencé en 2016 en prévision de l'accession au stade TA3-Ind.

2. Poursuivant les travaux initiés par FAST, nous avons proposé un nouveau framework pour répondre à la problématique de réutilisabilité des Fuzzy Extractors, originellement exhibée par Boyen (CCS 2004). Le premier Fuzzy Extractor, prouvé réutilisable, n'a été proposé que récemment (Canetti *et al.*, Eurocrypt 2016) et permet de gérer la métrique de Hamming. Cependant, dans le cas biométrique et à l'exception de l'Iris, cette distance n'est pas la plus adaptée alors que la différence d'ensembles qui permet de traiter des secrets sous forme d'ensembles est plus légitime dans le cas des empreintes digitales ou, comme nous le verrons, dans le cas du *fingerprinting* d'appareil. En proposant une instanciation de notre framework basé sur la différence d'ensembles, nous avons donc proposé le premier Fuzzy Extractor basé sur cette métrique à être réutilisable [5]. De plus, notre méthodologie, par l'usage d'un nouveau type de primitive (*pseudoentropic isometry*) présentera d'autres bénéfices que nous préciserons au cours du Chapitre 4.

3. Appliquer des Fuzzy Extractors à des données numériques souffrant d'une durée de vie plus courte que les données habituellement étudiées (biométrie, PUFs) nous a

conduit à définir les Fuzzy Extractors Adaptatifs (*Adaptive Fuzzy Extractors*) [5]. Un tel Fuzzy Extractor, permettra à un utilisateur de s'authentifier à partir d'un valeur d'authentification w' potentiellement très différente de sa valeur d'enrôlement w à condition que w' ait naturellement dérivé de w. Nous proposerons une méthodologie générique pour convertir tout Fuzzy Extractor réutilisable en un Fuzzy Extractor Adaptatif. Une fois encore, une instanciation, dans le cas de la différence d'ensembles, sera proposée.

4. Assimiler anonymement un utilisateur à un groupe d'amis et/ou d'appareils nous a conduit à l'étude de schémas de signature de groupe. D'autre part, puisque cet axe correspond au futur de la solution TA, nous nous sommes concentrés sur des solutions potentiellement résistantes à l'ordinateur quantique. Ces travaux ont donné lieu à deux différents schémas de signature de groupe basés sur la théorie des codes correcteurs [6, 7, 8]. En particulier, les travaux présentés dans [8] introduisent une nouvelle notion de preuve de connaissance désignée, *Testable weak Zero-Knowledge*. De telles preuves peuvent se voir comme des preuves à divulgation nulle de connaissance (ou *Zero-Knowledge*) pour lesquelles un vérifieur pourra tester si une information donnée consiste ou non en le secret du prouveur, sans n'apprendre rien d'autre que cette validité. Ces travaux, présentés au cours des Chapitres 5 et 6, ont été les premiers schémas de signature de groupe basés respectivement sur la métrique de Hamming et sur la métrique Rang.

Chapitre 1 :

Notations et Rappels Généraux

Contents

Organisée en cinq parties, ce chapitre présente les rappels nécessaires à une lecture auto-suffisante de ce manuscrit. Après une présentation des notations employées au long de ce manuscrit via la **Section 1.1**, nous proposons les sections suivantes :

- **Section 1.2.** Rappel sur les Codes Correcteurs ;
- **Section 1.3.** Rappels généraux sur la Cryptographie ;
- **Section 1.4.** Une introduction à la Cryptographie basée sur les codes correcteurs ;
- **Section 1.5.** Une introduction aux Fuzzy Extractors.

Si certaines notations plus spécifiques pourront apparaître au cours de la lecture, nous proposons de suivre les notations de la prochaine section tout au long de ce manuscrit.

1.1 Notation

Sous forme de tableau, nous proposons les définitions suivantes relatives aux domaines des mathématiques, des codes correcteurs d'erreurs et de la cryptographie.

Notation	Signification
$q = p^s$	Désigne puissance d'un nombre premier.
$\mathbb{F}_{q^m}$	Désigne le corps fini à $q^m = (p^s)^m$ éléments unique à isomorphisme près. En particulier, il est isomorphe à $(\mathbb{F}_q)^m$.
$\|$	Symbole de concaténation.
log	Fonction logarithmique définie en base 2.
$(\mathcal{M}, d)$	Espace métrique doté de la métrique d vérifiant les axiomes usuels (identité des indiscernables, symétrie, inégalité triangulaire).
$\mathcal{C}[n, k, d]$	Code linéaire de longueur n, dimension k et distance minimale d sur $\mathbb{F}_{q^m}$.
G	Matrice génératrice du code $\mathcal{C}$.
$A^\top$	Matrice transposée de la matrice A.
H	Matrice de parité du code $\mathcal{C}$.
$d_H(x_1, x_2)$	Distance de Hamming entre x_1 et x_2.
$\omega t(x)$	Poids du mot x. Hamming ou Rang en fonction du contexte.
$\mathcal{M}_{k\times n}(\mathbb{F}_{q^m})$	Ensemble des matrices à k lignes et n colonnes à coefficients dans $\mathbb{F}_{q^m}$.
S^n_ω	Ensemble des vecteurs appartenant à $\mathbb{F}_2^n$ de poids de Hamming égal à ω.
$[\![n]\!]$	Ensemble $\{1, \ldots, n\}$.
Σ_n	Ensemble des permutations sur $[\![n]\!]$.
$v[r]$	Pour une chaîne de caractère v (*respectivement* un vecteur), $v[r]$ correspond à la $r^{\text{ième}}$ coordonnée de v.
$\sqcup$	Désigne la réunion disjointe.
λ	Paramètre de sécurité d'un système.
$\mathcal{E}(I)$	L'entité $\mathcal{E}$ a la connaissance de l'information I.
$\mathcal{P}$ et $\mathcal{V}$	Prouveur et Vérifieur.
$\mathcal{A}(z : \mathcal{O})$	L'entité $\mathcal{A}$ a connaissance de l'information z et accès à l'oracle $\mathcal{O}$.
X	Si X est une variable, par abus de notation, on notera également X la distribution de probabilités associée.

$H(X)$	Entropie de la source X.
$H_\infty(X)$	Min-Entropie de la source X. $H_\infty(X) \stackrel{\text{def}}{=} -\log(\max_x \Pr[X = x])$.
$\tilde{H}_\infty(X)$	Min-Entropie conditionnelle de la source X.
$\mathbf{SD}(X, Y)$	*Statistical Distance* ou distance statistique des variables aléatoires A et B.
D	Un Distingueur D est un programme arbitraire qui retourne $b \in \{0, 1\}$. En pratique, étant donnés deux distributions et un élément, le distingueur décide de quelle distribution provient l'élément.
$\mathcal{D}$	Classe de distingueurs.
$\mathcal{D}_s$	Classe de distingueurs de taille au plus s.
$\delta^{\mathcal{D}_s}(X, Y)$	Distance calculatoire entre X et Y (voir Définition 40).
$X \approx_{s,\epsilon} Y$	Signifie que $\delta^{\mathcal{D}_s}(X, Y) \leqslant \epsilon$.
U_l	Distribution uniforme sur $\{0, 1\}^l$.
H	Désigne génériquement une fonction de hachage modélisée comme un oracle aléatoire. Ses paramètres dépendront du contexte.
ROM	Random Oracle Model ou Modèle de l'oracle aléatoire.

Table 1.1: Notations globales

1.2 Rappels sur les codes correcteurs

Les premiers codes correcteurs d'erreurs furent introduits par Richard Hamming dans les années 1940 puis par Claude Shannon qui introduisit la théorie de l'information. À cette époque, Hamming travaille sur un modèle de calculateur à carte perforée dont les nombreuses erreurs d'exécution étaient dépannées à la main par les ingénieurs. En l'absence de ces derniers, les machines se retrouvaient bloquées lors des périodes chômées. Hamming propose alors l'idée d'introduire de la redondance à la fin des mots pour pouvoir détecter les éventuelles erreurs : la théorie des codes était née. En parallèle, un de ses collègues chez Bell, Claude Shannon, formalisait la théorie de l'information comme une branche des mathématiques. Les codes correcteurs d'erreurs constituent un outil visant à améliorer la fiabilité des transmissions sur un canal bruité : par de la redondance, l'idée est de transmettre plus de données que la quantité d'information à transmettre. Lorsque les formes de redondance sont exploitables, il est possible de corriger les erreurs induites par le bruit du canal : c'est le décodage. De nos jours, les codes correcteurs se retrouvent

dans tous les domaines liés à l'information (télécommunications, mémoire RAM, CD, DVD, codes barres, QR codes, ...).

La majorité des rappels proposés ici sont tirés de la partie 7 de l'ouvrage *Mathématiques L3-Mathématiques appliquées* [9].

1.2.1 Premières définitions

Définition 1 (Code correcteur). *Soient un alphabet A et A^* l'ensemble des vecteurs formés à partir des éléments de A. Un code $\mathcal{C}$ est un sous-ensemble de vecteurs de A^*, dont tous les éléments sont appelés mot de code. Si $A = \{0, 1\}$, $\mathcal{C}$ est dit binaire.*

En général, les alphabets utilisés sont des corps finis ($A = \mathbb{F}_q$); c'est le cas dans cette thèse.

Définition 2 (Code linéaire). *Un code linéaire $\mathcal{C}$ sur $\mathbb{F}_q$ de longueur n et de dimension k est un sous-espace vectoriel de $\mathbb{F}_q^n$ de dimension k. Un tel code est noté $[n, k]_q$.*

Dans toute la suite, le terme espace ambiant réfère à $\mathbb{F}_q^n$et la longueur du code est la dimension de l'espace ambiant.

Définition 3 (Matrice génératrice). *Une matrice génératrice d'un code linéaire $\mathcal{C}$ est une matrice dont les vecteurs lignes forment une base de l'espace vectoriel $\mathcal{C}$.*

Encodage Soit $\mathcal{C}$ un code $[n, k]_q$. Un message m est considéré comme un élément de $\mathbb{F}_q^k$. L'encodage est donné par une application linéaire de $\mathbb{F}_q^k$ dans $\mathbb{F}_q^n$: l'image de x par cette application est un mot du code $\mathcal{C}$, l'encodage de x. Étant donnée une matrice génératrice G du code $\mathcal{C}$, l'encodage de x est donné par le produit matrice-vecteur xG.

Définition 4 (Produit sur $\mathbb{F}_q^n$). *Par analogie avec le produit scalaire euclidien usuel, on définit un produit sur $\mathbb{F}_q^n$. Pour $x, y \in \mathbb{F}_q^n$, on a*

$$x.y = \Sigma_{i=1}^{n} x_i y_i.$$

Définition 5 (Code dual). *Soit $\mathcal{C}$ un code $[n, k]_q$. On définit le dual $\mathcal{C}^\top$ de $\mathcal{C}$ par*

$$\mathcal{C}^\top = \{y \in \mathbb{F}_q^n | x.y = 0, \forall x \in \mathcal{C}\}.$$

Définition 6 (Matrice de parité). *Soit $\mathcal{C}$ un code $[n, k]_q$ de matrice génératrice G. On dit qu'une matrice H est une matrice de parité (ou matrice de contrôle) de $\mathcal{C}$ si H est une matrice génératrice du code dual $\mathcal{C}^\top$.*

Comme le code dual est également un espace vectoriel, il peut aussi avoir plusieurs matrices génératrices. Pour trouver le code dual, il suffit de résoudre le système $G.x^\top = 0$ pour $x = (x_1, \ldots, x_n)$. L'ensemble des x solutions forme le code dual.

Pour encoder un message, il peut être intéressant d'avoir une forme particulière pour la matrice génératrice, à savoir *la forme systématique.*

Définition 7 (Forme systématique)**.** *On dit qu'une matrice génératrice G d'un code est sous forme systématique quand $G = [I_k \| A]$, où I_k désigne la matrice identité $k \times k$ et A une matrice $k \times (n-k)$.*

Par abus de langage, on dit qu'un code est mis sous forme systématique s'il est donné par une matrice génératrice qui est sous forme systématique.

Proposition 1. *Quitte à permuter des colonnes, tout code $\mathcal{C}$ peut-être mis sous forme systématique.*

Proof. Une matrice génératrice d'un code $[n, k]$ étant une base du code, en tant qu'espace vectoriel, il existe nécessairement une sous-matrice $k \times k$ inversible dans cette matrice génératrice. Puisque des combinaisons linéaires des lignes de la matrice génératrice ne modifient pas le code, il existe donc une matrice génératrice du code comportant une sous-matrice identité $k \times k$. On peut alors se ramener à une forme systématique par permutation des colonnes. □

Distance de Hamming

Définition 8 (Poids de Hamming)**.** *Le poids de Hamming d'un mot x de $\mathbb{F}_q^n$ est le nombre de coordonnées non nulles de x. On le note $\omega t(x)$.*

On en déduit la distance de Hamming définie comme suit:

Définition 9 (Distance de Hamming)**.** *La distance de Hamming entre deux mots x et y est le nombre de lettres qui diffèrent entre x et y, c'est-à-dire*

$$d_H(x, y) = \omega t(x - y).$$

Il est clair que la distance de Hamming définit une distance sur $\mathbb{F}_q^n$.

Définition 10 (Distance minimale)**.** *La distance minimale d d'un code $\mathcal{C}$ est définie par*

$$d(\mathcal{C}) = min_{x,y \in \mathcal{C}, x \neq y} d_H(x, y).$$

On en déduit (assez) directement que $d(\mathcal{C}) = \min_{x \in \mathcal{C}, x \neq 0} \omega t(x)$.

Définition 11. *On appelle code $[n, k, d]_q$ un code $\mathcal{C}$ de longueur n, de dimension k et de dimension minimale d sur $\mathbb{F}_q$.*

Décodage Décoder dans un code $\mathcal{C}$ désigne l'action d'associer un mot du code à un mot de l'espace vectoriel. Soit c un mot du code $[n, k, d]_q$: en envoyant ce mot sur un canal; des erreurs peuvent apparaître et on reçoit un mot de l'espace ambiant $x = c + e$, où e représente le vecteur d'erreurs reçu. Tous les mots du code sont à une certaine distance du mot x. La question est de savoir, à partir de x, à quel mot transmis il peut correspondre. Si l'on suppose que l'erreur est faible, il est naturel de postuler que le mot duquel on est parti est le mot du code $\mathcal{C}$ le plus proche de x.

Dans les faits, trouver un algorithme de décodage efficace est un grand challenge de la théorie des codes. Souvent les codes sont structurés de façon à ce qu'il existe un algorithme de décodage efficace.

Décodage au maximum de vraisemblance Le décodage au maximum de vraisemblance est ainsi défini.

Définition 12. *Étant donné un code $\mathcal{C}$ de longueur n et un mot x de $\mathbb{F}_q^n$, on appelle décodage au maximum de vraisemblance de x le fait d'associer à x un mot du code $\mathcal{C}$ le plus proche de x pour la distance de Hamming. On note $\mathsf{Decode}(x, \mathcal{C})$ cette opération.*

$$\mathsf{Decode}(x, \mathcal{C}) = c \text{ tel que } d_H(x, y) = min_{c \in \mathcal{C}} d_H(c, y).$$

On a le théorème suivant au sujet du décodage au maximum de vraisemblance.

Théorème 1. *Soit $\mathcal{C}$ un code $[n, k, d]_q$. Il permet de corriger, par décodage à maximum de vraisemblance, au plus $t = \lfloor \frac{d-1}{2} \rfloor$ erreurs de manière univoque.*

Proof. Considérons les boules de rayon $t = \lfloor \frac{d-1}{2} \rfloor$ centrées sur les mots du code $\mathcal{C}^\top$. Ce t est le plus grand rayon tel que ces boules ne s'intersectent pas. Ainsi, à tout mot de l'espace inclus dans dans une des boules, on peut associer de manière univoque un mot du code: le centre de la boule. Ces mot de l'espace correspondent par définition exactement à tous les mots du code avec au plus t erreurs. □

Décodage par syndrome Le décodage par syndrome est une méthode générique de décodage des codes linéaires. Elle est optimale dans le sens où elle décode tout mot au maximum de vraisemblance: à tout mot de l'espace, on associe un mot de code le plus proche au sens de la distance de Hamming. Néanmoins, cette méthode souffre d'une complexité exponentielle et ne peut donc s'appliquer qu'à des codes de petites dimensions.

Définition 13 (Syndrome)**.** *Soit $\mathcal{C}$ un code $[n, k, d]_q$ de matrice de parité H. Le syndrome d'un mot x de $\mathbb{F}_q^n$ est le produit $s = H.x^\top \in \mathbb{F}_q^{n-k}$.*

Notons que si $x \in \mathcal{C}$, alors son syndrome est nul (voir Définition 6).

Ainsi, lorsque l'on utilise un code linéaire, avec G une matrice génératrice et m le mot à encoder, le mot encodé est donc mG. La transmission sur un canal bruité donne $x = mG + e$. Le syndrome de x, $H.x^\top = H(mG + e)^\top = He^\top$ ne dépend que de l'erreur.

Définition 14. *Soient $\mathcal{C}$ un code $[n, k, d]_q$ et $a \in \mathbb{F}_q^n$. On définit le translaté de a par $\mathcal{C}$ comme étant l'ensemble des mots $\{a + c | c \in \mathcal{C}\}$.*

On définit $\sim_\mathcal{C}$ la relation d'équivalence modulo un espace vectoriel $\mathcal{C}$ par

$$x \sim_\mathcal{C} y \iff x - c \in \mathcal{C}.$$

Le translaté de a par $\mathcal{C}$ est sa classe d'équivalence modulo $\mathcal{C}$, ce qui induit les propriétés suivantes.

Proposition 2. *Nous proposons les trois résultats suivants :*

1. *Deux mots sont dans le même translaté si et seulement si ils ont le même syndrome ;*
2. *Tous les translatés d'un code $\mathcal{C}$ ont le même nombre d'éléments $|\mathcal{C}|$;*
3. *Deux translatés sont soit disjoints soit égaux.*

Proof. On a $x \sim_\mathcal{C} y$ si et seulement si $x - y \in \mathcal{C}$; cela est équivalent à avoir $H.(x-y)^\top = 0$, c'est-à-direque x et y ont le même syndrome. Les deux propriétés suivantes viennent du fait que les translatés modulo $\mathcal{C}$ sont des classes d'équivalence pour $\sim_\mathcal{C}$. □

Définition 15. *À tout translaté, on peut associer un mot de plus petit poids contenu dans ledit translaté. Ce mot est appelé représentant principal. Lorsqu'il existe plusieurs choix pour ce mot, on en prend un quelconque.*

On peut alors créer une table de syndromes qui associe à chaque syndrome possible de $\mathbb{F}_q^{n-k}$ un représentant principal pour le translaté correspondant. Ainsi, le décodage par syndrome se déroule comme suit pour un code de matrice de parité H. Tout mot reçu x est corrigé en $x - cl(x)$, où $cl(x)$ désigne le représentant principal associé au syndrome $Hx^\top$ de x.

Théorème 2. *Le décodage par syndrome est un décodage à maximum de vraisemblance.*

Proof. Un tel décodage décode x en un mot de code $\mathcal{C}$ le plus proche de x. En effet, soit $c = x - cl(x) \in \mathcal{C}$ obtenu par décodage par syndrome de x et soit $c_1 \in \mathcal{C}$, alors $d_H(x, c) \leqslant d_H(x, c_1)$ car $d_H(x, c) = \omega t(x - c) = \omega t(cl(x)) \leqslant \omega t(x - c_1) = d_H(x, c_1)$ puisque $x - c$ et $x - c_1$ appartiennent au même translaté. □

1.2.2 Bornes sur les codes correcteurs

Nous nous intéressons désormais aux principales bornes pour les codes correcteurs. Considérons un code $\mathcal{C}$ de paramètres $[n, k, d]$. On appelle $A_q(n, d)$ (*respectivement* $B_q(n, d)$) le nombre maximum de mots que peut contenir un code (*respectivement* code linéaire) de paramètres $[n, k, d]$. Sauf indication contraire, $t = \lfloor \frac{d-1}{2} \rfloor$.

Borne de Hamming

Nous avons le théorème suivant.

Théorème 3 (Borne de Hamming)**.**

$$B_q(n, d) \leqslant A_q(n, d) \leqslant \frac{q^n}{\Sigma_{i=0}^{t} \binom{n}{i} (q-1)^i}$$

Proof. Soit un code $\mathcal{C}$ de longueur n et de distance d avec $A_q(n, d)$ mots. Les boules $B(c, t)$ centrées sur les mots c du code et de rayon t sont disjointes (car sinon il existerait deux mots à distance strictement inférieure à d) et donc la réunion de toutes les boules disjointes $B(c, t)$ est un ensemble d'élément de l'espace contenu dans $\mathbb{F}_q^n$. Les éléments d'une boule sont composés des mots à distance 0 du centre, ainsi que des mots à distance 1, jusqu'à la distance t. Le nombre de mots à distance 0 est 1. Le nombre de mots à distance 1 est $n(q-1) = \binom{1}{n}(q-1)$. Plus généralement, le nombre de mots à distance i vaut $\binom{i}{n}(q-1)^i$, le nombre de choix pour l'emplacement de l'erreur multiplié par le nombre de possibilités pour l'erreur en une coordonnée donnée. On obtient donc $A_q(n, d)$ boules disjointes avec le même nombre d'éléments, ce qui donne $A_q(n, d)\Sigma_{i=0}^{t} \binom{i}{n}(q-1)^i \leqslant q^n$. De plus, on a trivialement $B_q(n, d) \leqslant A_q(n, d)$, ce qui termine la preuve. □

Définition 16. *Un code qui vérifie l'égalité de cette borne est dit code parfait.*

En pratique, le fait qu'un code soit parfait entraîne que tous les mots de l'espace ambiant sont décodables de manière univoque, c'est-à-dire que tous les mots sont contenus dans la réunion des boules de rayon t centrées sur les mots du code.

Borne de Singleton

La borne de singleton donne une majoration de la distance minimale d'un code linéaire.

Théorème 4 (Borne de singleton)**.** *Soit $\mathcal{C}$ un code $[n, k, d]_q$. La distance minimale d du code $\mathcal{C}$ vérifie :*

$$d \leqslant n - k + 1.$$

Le nombre d'erreurs décodables est toujours inférieur à $\lfloor \frac{d-1}{2} \rfloor$. Un code qui atteint cette borne est dit MDS (*Maximum Distance Separable*).

Borne de Gilbert-Varshamov

Nous rappelons ici la borne de Gilbert-Varshamov (GV). Cette borne a été introduite par Gilbert et Varshamov de manière indépendante en 1952. Contrairement aux résultats précédents, ils s'agit d'une borne inférieure. Avant toute chose, nous rappelons la définition du rayon de recouvrement et le lemme suivant.

Définition 17 (Rayon de recouvrement)**.** *Le rayon de recouvrement $r_c(\mathcal{C})$ d'un code est le plus petit rayon r tel que la réunion des boules $B(c, r)$ (pour c parcourant tous les mots du code) recouvre tout l'espace.*

Lemme 1. *Soit $\mathcal{C}$ un code linéaire de longueur n, de distance minimale d et tel que $\mathcal{C}$ contient $A_q(n, d)$ mots. Le rayon de recouvrement de $\mathcal{C}$ est alors au plus $d - 1$.*

Proof. Supposons que $r_c(\mathcal{C}) \geqslant d$, alors il existe un mot x de l'espace $\mathbb{F}_q^n \backslash \mathcal{C}$ dont la distance à chaque mot de $\mathcal{C}$ est au moins d (sinon on aurait $r_c(\mathcal{C}) \leqslant d$). On peut alors construire le code $\mathcal{C}' = \mathcal{C} \cup \{x\}$. Ce code a une distance minimale au moins de d en raison de la condition sur le rayon de recouvrement et contient $|\mathcal{C}| = A_q(n, d) + 1$ mots. Or, comme $A_q(n, d)$ est maximal, on a par contradiction que $r_c(\mathcal{C}) \leqslant d - 1$ □

Nous établissons maintenant la borne de Gilbert-Varshamov.

Théorème 5. *Soit $\mathcal{C}$ un code longueur n et de distance minimale d. On alors :*

$$\frac{q^n}{\Sigma_{i=0}^{d-1} \binom{i}{n} (q-1)^i} \leqslant A_q(n, d).$$

1.2.3 Exemples de Codes

Tout en s'appuyant sur des résultats généraux (algorithmes de décodage, bornes et cryptographie basée sur les codes), les travaux présentés dans ce manuscrit s'appuient sur certaines familles de codes sans les manipuler précisément. Nous les citons à titre d'information et référons le lecteur à la littérature pour plus de détails :

- **BCH.** Introduits par Bose et Chaudhuri [10] et Hocquenghem [11], ils présentent un algorithme de décodage efficace (linéaire en n la longueur du code considéré). Ils sont utilisés dans les constructions de Fuzzy Extractors basés proposés dans [12, 13] ;

- **RS.** Les codes de Reed-Solomon (RS) constituent des codes pour lesquels ce sont des symboles plutôt que des bits qui sont considérés. Ainsi, l'algorithme de décodage de Welch-Berlekamp [14] permet de corriger l'apparition d'erreurs regroupées par paquets. Cet algorithme de décodage a également permet la conception de Fuzzy Extractors [15, 12, 13] ;
- **Goppa.** Bien qu'un distingueur ait récemment été exhibé [16], aucune attaque n'a été proposée. Les codes de Goppa [17], peu structurés, restent donc très utilisés dans le domaine de la cryptographie basée sur les codes (*e.g.* le voir le schéma CFS en section 1.4). Nous les utilisons implicitement au Chapitre 5 ;
- **LRPC.** Basés sur la métrique rang [18, 19], les *Low Rank Parity Check* codes [20] seront définis et utilisés au Chapitre 6.

1.3 Rappels sur la Cryptographie

La cryptologie au sens actuel du terme trouve ses fondements dans les travaux de Auguste Kerckhoffs [21] qui énonça les principes qu'un système cryptographique doit respecter pour assurer une communication confidentielle. La cryptologie s'est peu à peu détachée d'un statut militaire pour véritablement se développer en tant que science à part entière au cours du vingtième siècle et ce, pour deux principales raisons :

- Les guerres, notamment les guerres mondiales, ont poussé les états à investiguer de manière active et continue dans ces domaines. Un des exemples les plus célèbres est la conception de la machine ENIGMA utilisée par les allemands lors de la seconde guerre mondiale et dont le secret fut percé par les britanniques, ce qui, estime-t-on conféra un avantage certain dans l'issue du conflit.
- L'explosion des systèmes de communication, à la fin du vingtième siècle, ont permis des échanges électroniques dans plusieurs domaines majeurs (industriel, bancaire, commerce en ligne voire administration). Alors que la préoccupation première était l'interopérabilité de tels systèmes, il a vite fallu engagé des réflexions sécuritaires pour lesquelles la cryptographie constitue une brique élémentaire.

La cryptologie, étymologiquement *science du secret*, se partage selon deux domaines complémentaires: la cryptographie et la cryptanalyse. Pendant que les cryptographes élaborent des systèmes permettant d'assurer certains fondements sécuritaires, les cryptanalystes étudient leur robustesse en essayant de casser les algorithmes proposés par ces derniers.

Dans cette partie, nous présentons un certain nombre de notions et définitions cryptographiques usuelles que nous avons jugées utile de rappeler pour une lecture intelligible de nos travaux.

1.3.1 Sécurité Prouvée et Complexité de problèmes

Sécurité Prouvée

Il est généralement acquis que la sécurité prouvée a été introduite en 1984 par Goldwasser et Miccali [22]. Ce domaine permet d'analyser de manière formelle la sécurité d'une primitive cryptographique. Ainsi, il est nécessaire de définir précisément ce que le terme "sécurité" signifie pour un schéma donné. En fait, il faut donc précisément établir les objectifs que doit remplir le schéma étudié et quelles sont les attaques auxquelles il doit pouvoir résister.

Par exemple, un schéma de chiffrement a pour objectif de protéger la confidentialité d'un message. En fonction du contexte, certains scénarios d'attaques sont envisageables: on parle de *modèle de sécurité*. En général, un modèle de sécurité se définit à travers un jeu interactif entre un attaquant et un challengeur qui contrôle l'environnement de ce dernier. Pour gagner un jeu, un attaquant doit réaliser des tâches dont on veut généralement prouver qu'elles sont difficiles. Par exemple, dans le cas du chiffrement, l'attaquant peut avoir pour objectif de retrouver un clair à partir d'un chiffré donné.

En général, pour prouver la sécurité d'une primitive, l'idée est d'établir une réduction entre le modèle de sécurité et un problème jugé difficile. Plus précisément, gagner un jeu associé à un modèle de sécurité permettra alors de résoudre un problème difficile.

Modèle de l'oracle aléatoire Introduit par Bellare et Rogaway [23], le modèle de l'oracle aléatoire ou ROM (pour *Random Oracle Model*) suppose que toute fonction de hachage[1] se comporte comme une fonction aléatoire. Autrement dit, leurs sorties sont indistinguables de valeurs aléatoires. Ainsi, prouver la sécurité d'un schéma dans le ROM revient à remplacer toute fonction de hachage par un oracle aléatoire. À chaque appel, l'oracle retourne une sortie uniformément distribuée dans l'espace d'arrivée; évidemment sur une entrée déjà rencontrée l'oracle retourne la même sortie.

Modèle standard Contrairement au ROM, le modèle standard ne pose aucune hypothèse sur les fonctions de hachage utilisées. Les preuves de sécurité dans ce modèle sont donc généralement considérées comme plus fortes. En effet, certains travaux ont souligné que la sécurité dans le ROM pouvait conduire à des instanciations non sécurisées

[1] Nous définissons cette primitive en 1.3.2.

dans le modèle standard [24, 25][2].

La factorisation et le calcul du logarithme discret sont des problèmes considérés comme standards puisque largement éprouvés par la communauté, notamment depuis leur usages respectifs dans la conception des célèbres cryptosystèmes RSA [27] et ElGamal [28]. Ces problèmes sont donc considérés comme résistants aux technologies actuelles (Shorr a montré que ces deux problèmes pouvaient résolus en temps polynomial par l'ordinateur quantique [29]). Ainsi, proposer un cryptosystème dont la sécurité se réduirait à un tel problème reviendrait à prouver sa sécurité dans le modèle standard.

Complexité de problèmes

Notions de complexité Certains problèmes, comme la factorisation ou le logarithme discret, sont considérés difficiles car largement étudiés par la communauté sans qu'aucun algorithme efficace –c'est-à-dire en temps polynomial– de résolution n'ait été produit. D'autre part, certains problèmes jouissent d'assomptions de sécurité plus formelles. En cryptographie, on relie souvent la difficulté d'un problème à sa *classe de complexité*. Les définitions rappelées ici sont informelles pour la plupart. Le lecteur est invité à consulter des ouvrages adaptés tels que [30] pour plus de détails.

Avant d'introduire les classes de complexité, nous rappelons ce qu'est une machine de Turing en reprenant les explications données par E. Bresson dans son manuscrit de thèse [31]. Imaginé par Turing en 1936, le concept de machine de Turing permet de donner une définition précise au concept d'algorithme.

Définition 18 (Machine de Turing)**.** *Une machine de Turing est un triplet* (Σ, Q, δ) *vérifiant les propriétés suivantes:*

- Σ *est un ensemble fini non vide appelé alphabet et contenant un élément particulier, noté* I *;*
- Q *est un ensemble fini non vide, appelé ensemble des états.* Q *contient un élément particulier* q_0 *appelé état initial et un sous-ensemble* Q' *non vide dont les éléments sont appelés états finaux ;*
- δ *est une application de* $\Sigma \times Q$ *dans* $\{\leftarrow, -, \rightarrow\} \times \Sigma \times Q$*.*

Fonctionnement d'une machine de Turing Un tel concept se voit comme un automate comportant un ruban infini et une tête de lecture/écriture. Le ruban est composé de cases où la tête de lecture/écriture peut lire/écrire des symboles de Σ et se déplacer vers la gauche ou la droite du ruban. On initialise la machine dans l'état q_0, le

[2]Ces résultats sont à pondérer par l'intéressante discussion proposée par Koblitz et Menezes [26].

ruban ne contenant que des symboles I pour un nombre fini de cases et la tête est placé le plus à gauche de ces cases. Maintenant, en fonction du symbole σ lu sur la case courante et de l'état de la machine $q \in Q$ et en notant $(\vec{v}, \sigma', q') = \delta(\sigma, q)$, on déplace la tête de lecture comme indiqué par $\vec{v}$, on y écrit σ' et on met la machine dans l'état q'.

L'hypothèse de Church-Turing assure que tout ce que peut faire un ordinateur peut être réalisé par une machine de Turing de telle sorte qu'un algorithme peut être identifié à une machine de Turing. Une machine satisfaisant la Définition 18 est dite déterministe puisque pour une instance donnée, l'enchaînement des instructions mène aux mêmes transitions.

Machine de Turing non déterministe Une variante de la définition précédente consiste à considérer une machine dont la fonction d'état peut prendre plusieurs valeurs à la fois et peut donc exécuter plusieurs instructions en même temps. Ainsi, les calculs d'une machine de Turing déterministe forment une suite alors que ceux d'une machine non déterministe forment un arbre de calcul non linéaire dans lequel chaque chemin correspond à une suite de calculs possibles. Pour un problème donné, une machine de Turing non déterministe bénéficiera d'une capacité de résolution supérieure mais d'un point de vue de la théorie de l'information ces deux modèles sont équivalents : une machine non déterministe calcule exactement les mêmes fonctions qu'une machine non déterministe mais avec des temps de calcul différents.

Classes de complexité Nous définissons informellement les classes de complexité qui nous intéressent dans ce manuscrit en les reliant aux notions de machines de Turing évoquées ci-dessus. Rappelons d'autre part qu'un problème de décision est un problème pour lequel la réponse au problème est "oui" ou "non".

- **Classe P.** Contient les problèmes de décision qui peuvent être décidés par une machine de Turing déterministe en temps polynomial par rapport à sa taille d'entrée. C'est l'ensemble des problèmes *faisables* ou *faciles*.

- **Classe NP.** Ensemble des problèmes de la classe P Contient l'ensemble des problèmes qui peuvent être décidés par une machine de Turing non déterministe en temps polynomial par rapport à la taille de l'entrée. Intuitivement, cela revient à dire que l'on peut vérifier *rapidement* (*i.e.* par une machine de Turing déterministe en temps polynomial) si un candidat est bien solution d'une instance donnée. C'est le cas par exemple du problème du voyageur de commerce.

- **Classe NP-difficile (*NP-hard*).** Ensemble des problèmes au moins aussi difficiles que ceux de la classe NP. Plus précisément, un problème Q est dit NP-difficile si pour tout problème $Q' \in$ NP, il existe une réduction polynomiale de Q vers Q'.

- **Classe NP-complet.** Intersection des classes NP-difficile et NP.

Une des questions fondamentales de la théorie de la complexité est de savoir si P = NP, *i.e.* s'il existe des problèmes qui sont dans NP sans être dans P. Baser la sécurité d'un cryptosystème sur un problème NP-complet apporte un argument supplémentaire pour croire en sa robustesse puisque casser (en temps polynomial) ledit cryptosystème reviendrait à montrer que P = NP alors que ce problème reste ouvert après des décennies d'intenses recherches et que, pour beaucoup dans la communauté, il semblerait que P ≠ NP. La situation est résumée par la Figure 1.1

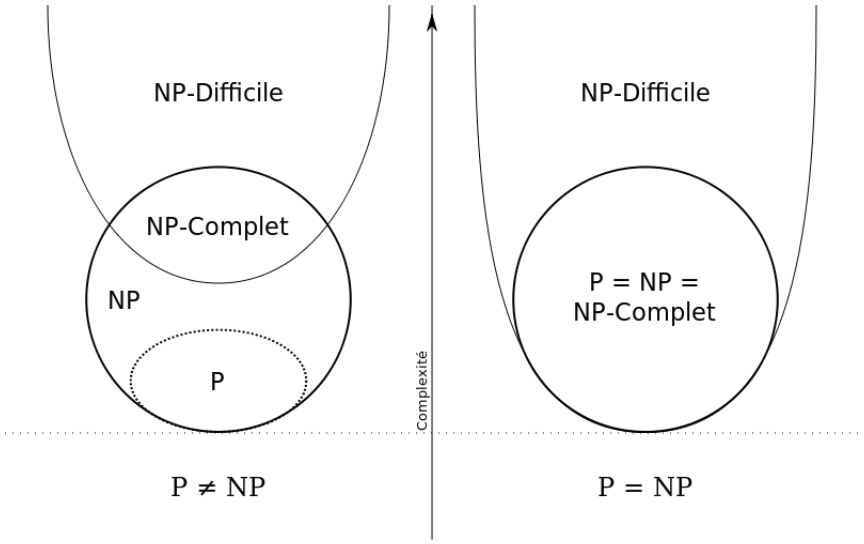

Figure 1.1: Représentation ensembliste des classes de complexité

1.3.2 Primitives Cryptographiques

Au vu du paragraphe précédent, il n'existe pas de définition permettant de dessiner les contours d'un problème difficile. Aussi, la communauté cryptographique proposa une définition heuristique. Afin de définir un problème difficile, nous nous appuyons sur la définition d'une fonction négligeable.

Définition 19 (Fonction négligeable et Problème Difficile)**.** *Une fonction $f : \mathbb{N} \to \mathcal{R}$ est dite négligeable si $\forall c \in \mathbb{N}, \exists k_0, \forall k \geqslant k_0 : |f(k)| < k^{-c}$. Un problème sera dit difficile s'il n'existe aucun algorithme le résolvant avec une probabilité non négligeable.*

Fonction à sens unique et fonction de hachage

La cryptographie s'appuie sur les fonctions à sens unique pour pour assurer la sécurité des protocoles. Une fonction à sens unique est une fonction facile à calculer mais difficile

à inverser. il est intéressant de noter que P $\neq$ NP impliquerait l'existence de fonctions à sens uniques telles que définies ci-dessous.

Définition 20 (Goldreich [32]). *Une fonction $f : \{0,1\}^* \to \{0,1\}^l$ est dite à sens unique si elle satisfait les deux condition suivantes :*

- *facile à calculer : il existe un algorithme polynomial $\mathcal{A}$ qui, pour tout $x \in \{0,1\}^*$ retourne $f(x)$;*
- *inverser f est un problème difficile: autrement dit, pour tout algorithme polynomial $\mathcal{A}'$ et pour tout l suffisamment grand, si $x \leftarrow U_l$, alors :*

$$Pr[\mathcal{A}'(f(x), 1^l) \in f^{-}1(f(x))] < \frac{1}{poly(l)}$$

En pratique, et conformément à l'incertitude P = NP?, on s'appuie sur une définition heuristique pour les fonctions à sens unique en considérant comme difficile tout problème pour lequel on ne connaît que des algorithmes de résolution *super-polynomiaux* c'est-à-dire en $n^{\omega(1)}$.

On définit alors une fonction à trappe f comme étant une fonction à sens unique dans le cas général mais pour laquelle il existe une information trappe permettant d'inverser f. On propose la définition suivante:

Définition 21 (Fonction à trappe). *Une fonction $f : \{0,1\}^* \to \{0,1\}^l$ est dite à trappe si elle satisfait les condition suivantes :*

- *facile à calculer : il existe un algorithme polynomial $\mathcal{A}$ qui, pour tout $x \in \{0,1\}^*$ retourne $f(x)$;*
- *en général, inverser f est un problème difficile: autrement dit, pour tout algorithme polynomial $\mathcal{A}'$ et pour tout l suffisamment grand, si $x \leftarrow U_l$, alors :*

$$Pr[\mathcal{A}'(f(x), 1^l) \in f^{-}1(f(x))] < \frac{1}{poly(l)}$$

- *il existe une trappe : autrement dit, une certaine valeur $trappe \in \{0,1\}^*$ qui, pour tout algorithme A', permet d'inverser f :*

$$\exists trappe \in \{0,1\}^*,\ Pr[\mathcal{A}'(f(x), 1^l, trappe) \in f^{-}1(f(x))] = 1.$$

Une fonction résistante aux collisions est une fonction pour laquelle il est un problème difficile de trouver une collision. Plus formellement, nous avons la définition suivante.

Définition 22. *Une fonction* $f : \{0,1\}^* \to \{0,1\}^l$ *est dite résistante aux collisions si trouver deux éléments de* $\{0,1\}^*$ *qui ont la même image est un problème difficile; autrement dit :*

$$\mathsf{Pr}[\mathcal{A}(x \in \{0,1\}^*, x' \in \{0,1\}^* : x \neq x' \wedge f(x) = f(x'))] < \frac{1}{\mathsf{poly}(l)}$$

Une fonction de hachage cryptographique est une fonction à sens unique particulière: elle permet de compresser des données d'un ensemble de définition $\mathcal{D}_\lambda$ vers un ensemble de sortie $\mathcal{S}_\lambda$ dont la taille fixée est régie par un paramètre de sécurité. Nous proposons la définition suivante :

Définition 23. *On dit que* h *est une fonction de hachage cryptographiquement sure si elle satisfait les trois propriétés suivantes :*

résistance à la préimage : h *est une fonction à sens unique;*

résistance à la seconde préimage : étant donné une message $m \in \mathcal{D}_\lambda$*, il doit être un problème difficile (Définition 19) de trouver* $m' \neq m$ *tel que* $h(m') = h(m)$*;*

résistance aux collisions : h *est une fonction résistante aux collisions (Définition 22).*

L'existence (heuristique) de fonctions à trappe permettent de construire des systèmes de chiffrement et de signature. Par exemple, en prenant le cas chiffrement RSA [27], la fonction à trappe est la procédure de chiffrement alors que la trappe trappe consiste en la clé privée e. Après avoir chiffré, il est calculatoirement difficile de retrouver le message originel sans la connaissance de trappe.

PRF, Extracteurs et KDF

Fonctions pseudo aléatoire ou Pseudo-Random Functions (PRF) En 1984, Goldreich *et al.* proposent de formaliser le comportement aléatoire des fonctions en introduisant les famille de fonction pseudo-aléatoire, ou Pseudo-Random Function Family (PRF) [33]. L'ensemble des fonctions de X vers Y est noté Y^X.

Définition 24 (Informelle)**.** *On dit que* $\mathcal{F} \subset Y^X$ *est une PRF si aucun adversaire ne peut distinguer une fonction aléatoirement choisie dans* $\mathcal{F}$ *d'une fonction aléatoirement choisie dans* Y^X*.*

L'idée derrière les PRFs est de simuler le fonctionnement d'un oracle aléatoire : à tout élément d'un ensemble de départ, la PRF assignera de manière aléatoire une valeur dans l'espace d'arrivée.

Extracteur fort ou Strong Extractor Intuitivement, un extracteur fort[3] est une fonction qui prend en entrées une donnée faiblement aléatoire (*i.e.* loin d'être uniformément distribuée) et une graine publique, uniformément distribuée elle, pour retourner une chaîne binaire uniformément distribuée et donc apte à un usage cryptographique. L'idée est d'alors d'utiliser l'aléa public (la graine) pour extraire l'entropie contenue dans la donnée privée faiblement aléatoire.

Avant de rappeler une définition formelle, nous rappelons la définition de distance statistique entre deux variables aléatoires.

Définition 25 (Distance Statistique)**.** *Soient X et Y deux distributions de probabilités. La distance statistique X et Y est* $\mathbf{SD}(X,Y) \stackrel{def}{=} \frac{1}{2}\Sigma_v|Pr(X=v) - Pr(Y=v)|$.

Tel qu'introduit par Nisan et Zuckerman [34], un extracteur satisfait la définition suivante.

Définition 26 (Extracteur fort)**.** *Soit $\mathsf{Ext} : \{0,1\}^n \to \{0,1\}^l$ une fonction probabiliste en temps polynomial qui utilise r bits d'aléa. On dit que Ext est un (n,m,l,ϵ)-extracteur fort si pour toute distribution de probabilités W sur $\{0,1\}^n$ de min entropie m, on a $\mathbf{SD}((\mathsf{Ext}(W;X),X),(U_l,X)) \leqslant \epsilon$, où X est uniforme sur $\{0,1\}^r$.*

Originellement étudiés d'un point de vue statistique [34, 35, 36], des études plus récentes ont étudié le cas calculatoire où la distance statistique est remplacée par une distance calculatoire [37, 38].

Fonction de dérivation de clés (KDF) Une fonction de dérivation de clés ou KDF pour *Key Derivation Function* est une fonction qui dérive une ou plusieurs clés secrètes d'une valeur secrète non nécessairement adaptée à un usage cryptographique (*e.g.* mots de passe, phrases secrètes). Bien que les KDFs jouissent d'un usage répandu en pratique, ce n'est qu'en 2010 qu'une formalisation d'un tel outil a été proposée par Krawczyk [37]. Krawczyk y fait également le lien entre théorie et pratique en proposant une instanciation de ses définitions basée sur la construction HMAC [39] . Le modèle proposé suit la méthodologie "extrait-puis-étend" ou "extract-then-expand" de sorte qu'une KDF pourra se voir comme l'enchaînement d'un extracteur fort (calculatoire) et d'une PRF. Alors que l'extracteur permettra d'extraire l'entropie de l'entrée pour retourner une chaîne uniformément distribuée, la PRF, modélisant un oracle aléatoire, permettra alors de générer des clés supplémentaires à partir de cette chaîne uniformément distribuée.

Nous proposons la remarque suivante à des fins de rédaction.

Remarque 1. *Dans ses travaux sur l'extraction et la dérivation de clés [37, Section 5], Krawczyk a justifié que les oracles aléatoires modélisent des extracteurs calculatoires.*

[3] Le terme *fort* est souvent omis.

Le chiffrement

La transmission d'information sensible est un problème intemporel : seul un destinataire autorisé doit être en mesure d'avoir accès à l'information. Le chiffrement permet ainsi la transmission d'un message entre deux entités à travers un canal de sorte que les informations échangées apparaissent inintelligibles aux yeux de n'importe quelle entité autre que le destinataire désigné. Si le chiffrement symétrique existe depuis l'Antiquité (*e.g.* Chiffrement de César), le chiffrement asymétrique a été récemment proposé conjointement par Diffie et Hellman en 1976 [40] et par Merkle dans des travaux publiés à posteriori.

Chiffrement symétrique Émetteur et destinataire possèdent la même clé secrète qui va permettre à la fois de chiffrer et déchiffrer des messages. Les chiffrements symétriques modernes (TDES [41], AES [42]) possèdent l'avantage d'être rapides d'exécution mais posent les problèmes de transmission des clés secrètes sur des canaux non sûrs à priori et du grand nombre de clés à gérer (nombre de clés linéaires avec le nombre d'interlocuteurs pour un utilisateur donné).

Chiffrement Asymétrique Deux clés sont nécessaires : une clé publique pour chiffrer les messages vers le destinataire associé à ladite clé, et la clé privée permettant au destinataire de déchiffrer les messages reçus. Chaque utilisateur possède alors une seule paire de clés et la clé secrète n'a pas vocation à être transmise répondant ainsi aux limitations soulevées dans le cas du chiffrement symétrique. La sécurité d'un tel schéma repose sur le fait que la connaissance d'une clé publique pk_A associée à un utilisateur A ne doit pas permettre de retrouver sa clé privée associée sk_A. C'est ici qu'interviennent les fonctions à trappe où la trappe consiste alors en sk_A.

Ces systèmes de chiffrement souffrent en général de temps d'exécution plus longs que les systèmes symétriques de sorte qu'une utilisation conjointe en est souvent faite. L'idée est, pour deux utilisateurs donnés, d'utiliser la cryptographie à clé publique pour s'échanger un clé secrète qui leur permettra de chiffrer et déchiffrer efficacement leurs échanges consécutifs : c'est le principe du protocole TLS.

Plus formellement, nous définissons un schéma de chiffrement comme suit.

Définition 27. *Un schéma de chiffrement symétrique (respectivement asymétrique) est décrit par les quatre algorithmes suivants :*

- $\mathsf{Setup}(1^\lambda)$*, génère* param *les paramètres globaux du système à partir du paramètre de sécurité* λ *;*
- $\mathsf{KeyGen}(\mathsf{param})$*, génère la clé* k *(respectivement la clé publique* pk *et la clé privée* sk*).*

Les algorithmes Setup *et* KeyGen *pourront éventuellement être décrits de manière confondue;*

- $\mathsf{Enc}(\mathsf{k}, m; \mu)$ *(resp.* $\mathsf{Enc}(\mathsf{pk}, m; \mu)$*) qui génére le chiffré* c *à partir du message* m*, sous la clé de chiffrement* k *(resp.* pk*) et de l'aléa* μ*;*
- $\mathsf{Dec}(\mathsf{k}, c)$ *(resp.* $\mathsf{Dec}(\mathsf{sk}, c)$*) qui retourne le message originel (ou clair)* m *à partir du chiffré* c *ou* $\perp$*.*

Dans le paradigme des cryptosystèmes asymétriques, deux modèles de sécurité sont communément considérés : le modèle CPA pour *Chosen Plaintext Attack* et le modèle CCA pour *Chosen Ciphertext Attack*. Dans les deux cas, les jeux définissant les modèles de sécurité associés consistent pour un adversaire ayant connaissance de la clé publique d'un challengeur à tenter d'apprendre de l'information sur un chiffré que ce dernier aura généré à l'aide de sa clé privée. Dans le modèle CCA, l'adversaire a en plus l'accès à un oracle qui lui retournera les clairs associés à des chiffrés de son choix, exception faite du chiffré sur lequel il sera challengé. Pour plus de détails à ce sujet, nous invitons le lecteur à consulter les travaux présents dans [43] et les références qui s'y trouvent.

Dans le cas symétrique, la situation est un peu différente puisque clé de chiffrement et de déchiffrement ne constituent qu'une seule et même clé. L'adversaire ne pourra donc pas chiffrer autant de messages qu'il le désire. Bellare *et al.* ont alors proposé des modèles de sécurité adaptés au cas symétrique. Une notion équivalente à la sécuité CPA et à laquelle nous nous référerons au Chapitre 4 est le modèle de sécurité FTG-CPA (*Find-Then-Guess*) [1]. Afin d'obtenir plusieurs chiffrés de clairs de son choix, l'adversaire aura accès à un oracle de chiffrement. La définition proposée reprend celle de Bellare *et al.* [1]. La seule différence est que l'adversaire considéré ici sera un distingueur limité en taille plutôt qu'en temps de complexité. Cette modification est apportée pour permettre une harmonisation avec les définitions de Fuzzy Extractors que nous utiliserons au Chapitre 4.

Définition 28. *Soient* $\mathsf{Chif} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$*, un schéma de chiffrement symétrique (Définition 27) et* $\mathcal{O}^{encrypt(\mathsf{k},.)}$ *l'oracle qui retourne le chiffré, sous la clé* k*, de tout message* m*. Soit* D *un distingueur pouvant faire* q *appels à l'oracle* $\mathcal{O}^{encrypt}$*.*

On définit alors le jeu suivant entre Chif *et le distingueur* D

Experiment $\mathrm{Exp}^{\text{FTG-CPA-b}}_{\mathsf{Chif},D}(\lambda)$
1. $K \leftarrow \mathsf{KeyGen}(1^\lambda)$
2. $(m_0, m_1) \leftarrow D(\mathcal{O}^{\text{encrypt}})$
3. $c_b \leftarrow \mathsf{Enc}(\mathsf{k}, m_b)$
4. $b' \leftarrow D(c_b : \mathcal{O}^{\text{encrypt}})$
5. Return b'.

Figure 1.2: Jeu pour la sécurité FTG-CPA [1]

On dira que **Chif** *est de sécurité* $(\epsilon_{CPA}, s_{CPA})$ *si pour tout* $D \in \mathcal{D}^{s_{CPA}}$*, on a :* $Adv^{FTG-CPA}_{\mathsf{Chif},D}(\lambda) \stackrel{def}{=} Pr[D(Exp^{FTG-CPA-1}_{\mathsf{Chif},D}(\lambda)) = 1] - Pr[D(Exp^{FTG-CPA-0}_{\mathsf{Chif},D}(\lambda)) = 1] \leqslant \epsilon_{CPA}$.

Code d'authentification de message ou MAC

Alors que le chiffrement permet de protéger les messages échangés en confidentialité, il faudra également que le destinataire soit certain de la provenance et de l'intégrité des données reçues. En effet, si un adversaire venait à les modifier, le destinataire –en déchiffrant– ne serait pas en mesure de retrouver l'information qui lui était initialement destinée. Adaptés à la cryptographie à clé secrète, les MACs, pour *Message Authentication Codes*, répondent à ces objectifs. À partir de données à authentifier et d'une clé secrète, un MAC va générer un *tag* assurant la validité des données. Un algorithme de vérification permet alors de s'assurer que ledit tag est bien associée aux données transmises. La sécurité repose sur le fait qu'il est difficile pour un adversaire de générer un tag associé à des données sans connaître la clé secrète du MAC. Il existe de nombreuses spécifications permettant de générer des MACs à partir de schémas de chiffrement ou de fonctions de hachage [44, 45, 46, 39].

Dans un contexte à clé publique, la signature numérique que nous introduisons au cours de la prochaine sous-section peut se voir comme l'analogue au MAC.

1.3.3 Authentification et Signature

Protocoles de reconnaissance

En 1987, Fiat et Shamir [47] ont classifié les différentes catégories de protocoles de reconnaissance selon trois différents niveaux; En supposant, qu'Alice désire prouver son identité à Bob, nous avons :

1. **Authentification.** Alice peut prouver à Bob qu'elle est Alice mais une tierce entité ne pourra pas prouver à Bob qu'elle est Alice ;

2. **Identification.** Alice peut prouver à Bob qu'elle est Alice, et Bob ne pourra pas faire croire à quelqu'un d'autre qu'il est Alice ;

3. **Signature.** Ce protocole a les mêmes propriétés que le précédent et en plus, Alice ne peut pas dire qu'elle n'a pas prouvé son identité à Bob ou que le message qu'elle a signé était différent que celui reçu par Bob (*i.e.* elle ne peut pas répudier sa signature). Autrement dit, Bob ne peut pas faire croire qu'Alice lui a prouvé son identité alors qu'elle ne l'a pas fait.

En fait le protocole d'authentification au sens de [47] ne sera utilisé que lorsqu'Alice et Bob ont des intérêts communs. Dans le cas contraire, les deux autres protocoles

doivent être utilisés. En effet, dans le premier cas, Bob serait capable de prouver à une tierce entité qu'il est Alice alors que cela lui est impossible dans le cas d'un schéma d'identification. La nuance entre identification et signature est plus subtile : dans le cas du schéma d'identification, Bob est potentiellement capable de simuler des échanges entre Alice et lui-même pour faire croire qu'il a reçu un message d'Alice (comme s'il imitait sa signature), tandis qu'un protocole de signature l'empêche.

Dans la suite de cette thèse, comme dans de nombreux travaux actuels, nous confondons schémas d'authentification et d'identification. En effet, un protocole d'authentification au sens de [47] suppose d'une part qu'Alice et Bob ont des intérêts communs et que d'autre part, personne ne peut écouter leurs échanges (sinon ce dernier se retrouve dans les mêmes conditions que Bob et pourra donc se faire passer pour Alice) : cette confidentialité physique n'est pas assurée en général.

Schémas de Signature

En introduisant la cryptographie à clé publique [40], Diffie et Hellman ont aussi proposé le principe de la signature numérique. Par analogie avec la signature manuscrite, la signature numérique permet d'authentifier l'auteur d'un document. Ainsi, on requiert généralement qu'une signature soit :

- *Authentique.* L'identité du signataire doit être retrouvée.
- *Infalsifiable.* Personne ne peut se faire passer pour un autre.
- *Non-réutilisable.* La signature fait partie du document signé et ne peut être utilisée sur un autre document.
- *Inaltérable.* Une fois qu'un document est signé, il ne peut plus être modifié.
- *Non répudiable.* Le signataire ne peut pas nier avoir signé.

Pour résumer, un schéma de signature électronique permet donc à un signataire de produire une preuve qu'il a effectivement émis un message. En s'appuyant sur [48], nous rappelons la définition suivante.

Définition 29. *Un schéma de signature peut se décrire selon quatre algorithmes* $(\mathsf{Setup}_{\mathcal{S}}, \mathsf{KeyGen}_{\mathcal{S}}, \mathsf{Sign}_{\mathcal{S}}, \mathsf{Verif}_{\mathcal{S}})$*:*

- $\mathsf{Setup}_{\mathcal{S}}(1^{\lambda})$*, génère les paramètres globaux au système* param*. Il est parfois confondu avec l'algorithme* $\mathsf{KeyGen}_{\mathcal{S}}$*.*
- $\mathsf{KeyGen}_{\mathcal{S}}(\mathsf{param})$*, retourne une paire* $(\mathsf{sk}, \mathsf{vk})$*, où* sk *est la clé (secrète) de signature, et* pk *la clé de (publique) de vérification.*

- $\mathsf{Sign}(\mathsf{sk}, m; \mu)$, *retourne une signature* σ *du message* m*, sous la clé secrète* sk *et l'aléa* μ*.*
- $\mathsf{Verif}(\mathsf{vk}, m, \sigma)$, *vérifie la validité de la signature* σ *par rapport au message* m *et à la clé de vérification* vk*.*

Signature de groupe Chaum et van Heyst ont proposé le concept de signature de groupe en 1991 [49]. L'idée d'un tel protocole est de permettre à un groupe d'utilisateurs de signer des messages au nom du groupe. Plus précisément, au lieu de signer en son nom propre, chaque membre du groupe devient anonyme au sein dudit groupe et engage une responsabilité collective lorsqu'il signe des messages. Quiconque vérifiant la validité d'une signature sera convaincu que le signataire était effectivement un membre du groupe associé à une clé publique commune. Dans le cadre du projet TA, nous nous sommes intéressés aux schémas de signature de groupe pour répondre à une problématique de respect de la vie privée des utilisateurs dans le contexte de l'Internet des Objets. Les résultats de ces travaux seront exposés aux chapitres 5 et 6. Définitions formelles et modèle de sécurité adaptés y seront proposés.

Schéma d'engagement

Un schéma d'engagement (formalisé dans [50]) permet à un utilisateur d'engager une valeur sans la révéler mais sans qu'il n'ait la possibilité de modifier cette valeur par la suite. Informellement, un tel schéma fonctionne comme suit :

- $\mathsf{Setup}(1^\lambda)$ génère les paramètres du système à partir du paramètre de sécurité ;
- $\mathsf{Commit}(m; r)$ produit un engagement c à partir du message m en utilisant l'aléa r ;
- $\mathsf{Decommit}(c, m; r)$ débloque l'engagement c et révèle le message m ainsi qu'un témoin ω qui prouve l'ouverture correcte du message.

Un tel schéma doit vérifier deux notions de sécurité dite d'indistinguabilité et d'engagement (respectivement *hiding* et *binding*, en anglais). Ces deux propriétés assurent qu'un engagement c ne révèle pas d'information sur le message m (hiding) et que l'utilisateur ne pourra plus changer la valeur de m une fois l'engagement réalisé (binding).

Preuves de connaissances et preuves d'appartenance

Un protocole interactif peut se modéliser par un couple de machines de Turing $(\mathcal{P}, \mathcal{V})$ actives à tour de rôle qui communiquent à travers leurs rubans. On parlera de *preuves interactives* lorsqu'un un prouveur $\mathcal{P}$ essaie de convaincre un vérifieur $\mathcal{V}$ d'un certain théorème. Suivant la nature du théorème, on distingue deux types de preuves interactives:

- *preuves d'appartenance à un langage.* Le prouveur démontre qu'une donnée publique vérifie une certaine condition, autrement dit qu'elle appartient à un certain langage formel.

- *preuve de connaissance.* Le prouveur démontre sa connaissance d'un secret, non accessible au vérifieur.

Abordé en 1978 par Rabin dans [51], le concept de preuve de connaissance (*proof of knowledge*) a été ensuite repris par plusieurs travaux concurrents et/ou complémentaires [47, 52, 50, 53, 54]. À la fin d'un tel protocole, le vérifieur obtient une preuve que le prouveur connaît bien le secret sans que ce dernier ne soit jamais révélé.

Cependant, ce sont Goldwasser, Micali et Rackoff [55] qui, en 1985, ont été les premiers à quantifier l'information transmise lors d'un protocole interactif en introduisant le concept de divulgation nulle de connaissance (*zero-knowledge*). Intuitivement, l'idée est de montrer que toute quantité d'information calculable par le vérifieur après exécution du protocole est calculable en temps polynomial à partir de la connaissance seule des données initiales; autrement dit, la connaissance de la preuve n'apporte aucune information. Néanmoins, dans leur travaux, il est question de preuves d'appartenance à un langage plutôt que de preuve de connaissance. En pratique, la principale différence entre ces deux concepts est que dans le second cas, le prouveur est une machine de Turing (probabiliste) de puissance polynomiale et non infinie ; le vérifieur, quant à lui, est toujours une machine de Turing polynomiale. Ainsi, dans le cas des preuves de connaissance, en considérant que le prouveur connaît le témoin (*i.e.* son secret) d'une instance publique d'un problème difficile (Définition 19) alors l'acceptation du protocole assure au vérifieur que le prouveur, puisque polynomial connaît effectivement le secret.

Feige, Fiat et Shamir proposent alors de formaliser le concept de preuves de connaissance *zero-knowledge* [52] pendant que Brassard, Chaum et Crépeau proposent le concept voisin de preuves à divulgation minimum (*minimum disclosure proof*) [50].

Nous nous appuyons sur le manuscrit de E. Bresson [31] pour définir preuve de connaissance et propriété de divulgation nulle (dans toute la suite, on utilisera la terminologie anglaise *zero-knowlegde*).

Preuve de connaissance Lors d'une preuve de connaissance, le prouveur cherche à convaincre le vérifieur qu'il connaît une solution à un certain problème difficile. Cette solution, *i.e.* le secret du prouveur, n'est connue que de ce dernier. Une preuve d'appartenance à un langage peut donc se voir comme un preuve de connaissance au cours de laquelle $\mathcal{P}$ prouve sa connaissance d'un témoin ω pour le prédicat $x \in \mathcal{L}$. La principale différence entre preuve d'appartenance à un langage [55] et preuve de connaissance [47, 52, 50] est que dans le premier cas, $\mathcal{P}$ de puissance calculatoire infinie et

est donc capable de calculer son secret (ω) alors que dans le second cas, $\mathcal{P}$, de puissance polynomiale connaît son secret dès le début du protocole. Ainsi, lorsque le calcul d'un témoin est un problème difficile, le prouveur polynomial convainc donc le vérifieur qu'il connaît effectivement le secret.

Extracteur de connaissance ou *knowledge extractor* Une preuve de connaissance convainc un vérifieur que le prouveur $\mathcal{P}$ connaît effectivement un secret. "Connaître un secret" est alors formalisé par la notion d'extracteur (ou *knowledge extractor*). C'est une machine de Turing polynomiale disposant ayant le contrôle de $\mathcal{P}$: il pourra alors lui faire exécuter le protocole considéré un nombre polynomial de fois tout en contrôlant son ruban aléatoire. Noté $\mathcal{E}$, un tel extracteur est effectif si pour tout prouveur $\mathcal{P}$ accepté par le vérifieur $\mathcal{V}$ avec probabilité non-négligeable, $\mathcal{E}$ pourra calculer le secret à la suite de son interaction avec $\mathcal{P}$.

Définition Soit T un prédicat à deux variables $T(.,.)$ D et ω. On dit que ω est un témoin associé à D si $T(\omega, \mathsf{D}) = 1$. On considère une interaction entre $\mathcal{P}$ et $\mathcal{V}$, où D est une donnée initiale commune, au cours de laquelle $\mathcal{P}$ prouve la connaissance d'un témoin ω associé à D.

Définition 30 (Preuve de connaissance)**.** *Un protocole interactif, $(\mathcal{P}, \mathcal{V})$, est une preuve de connaissance pour un témoin associé à D si les conditions suivantes sont respectées:*

- ***Efficacité.*** *$\mathcal{P}$ et $\mathcal{V}$ sont des machines de Turing polynomiales;*
- ***Consistance (correctness).*** *Un prouveur honnête est accepté par un vérifieur honnête avec une probabilité écrasante :*

  $\forall D, \forall \omega, T(\omega, D) = 1 \rightarrow \mathsf{Pr}(\mathcal{P}(\omega), \mathcal{V})(x) = \mathsf{Accept}] > 1 - \mathsf{negl}(|D|)$.
- ***Significativité (soundness).*** *Il existe un extracteur de connaissance :*

  $\exists \mathcal{E}_{\mathcal{P}}, \forall D, \mathsf{Pr}(\mathcal{P}(*), \mathcal{V})(x) = \mathsf{Accept}] > 1 - \mathsf{negl}(|D|) \implies \mathsf{Pr}(T(\mathcal{E}_{\mathcal{P}}(D), D) = 1] > 1 - \mathsf{negl'}(|D|)$.

On dit alors que le prédicat T admet un système de preuve interactive de connaissance.

Preuves à divulgation nulle de connaissance ou *zero-knowledge*

La propriété de consistance assure le bon fonctionnement du système dans le cas d'intervenants $\mathcal{P}$ et $\mathcal{V}$ honnêtes alors que la significativité assure la sécurité du vérifieur face à un prouveur malhonnête. Un tel prouveur a une chance négligeable d'être accepté de sorte que $\mathcal{V}$ pourra faire confiance au protocole.

Il reste désormais à étudier le cas d'un vérifieur malhonnête face à un prouveur honnête. La question est de définir la sécurité du prouveur : pour qu'un tel protocole fonctionne, le prouveur doit convaincre le vérifieur. Pour conserver cette capacité, il doit laisser transparaître le moins d'information possible au sujet de son secret lors de l'exécution d'une preuve. C'est précisément pour répondre à cette problématique que la notion de zero-knowledge a été introduite dans [55]. L'objectif est d'assurer une exécution au cours de laquelle le vérifieur n'apprendra rien de plus que les données communes D.

Pour formaliser cette propriété en s'appuyant sur [55], on introduit un simulateur $\mathcal{S}$ qui est une machine de Turing probabiliste et polynomiale communicant avec $\mathcal{V}$. On dira que $\mathcal{S}$ est bon si sa sortie aléatoire formée par les échanges avec $\mathcal{V}$ et d'éventuelles données initiales est proche de la sortie aléatoire obtenue lors d'une interaction réelle entre $\mathcal{P}$ et $\mathcal{V}$. À partir de la donnée initiale D, on considère une interaction entre $\mathcal{P}$ et $\mathcal{V}$ au cours de laquelle $\mathcal{P}$ veut prouver qu'il connaît le secret ω associé à D. On appelle $\mathsf{Vue}(P(\omega))(\mathsf{D})$ la *vue* du vérifieur, c'est-à-direla distribution de variables aléatoires décrivant de tels échanges (les probabilités associées sont dues aux machines probabilistes $\mathcal{P}$ et $\mathcal{V}$).

Définition 31 (Zero-Knowledge)**.** *Le protocole interactif $(\mathcal{P}, \mathcal{V})$ est à divulgation nulle (zero-knowledge ou ZK) si pour tout vérifieur (possiblement malhonnête) $\tilde{\mathcal{V}}$, il existe un simulateur pro$\mathcal{S}_{\tilde{\mathcal{V}}}$ tel que :*

$$\mathsf{Vue}(P(\omega))(D) \sim \mathcal{S}_{\tilde{\mathcal{V}}}(D),$$

où le symbole $\sim$ peut signifier égalité parfaite, statistique ou calculatoire. On parle alors de protocole respectivement parfaitement, statistiquement ou calculatoirement zero-knowlegde.

Preuves à témoin indistinguable Les protocoles à divulgation nulle sont souvent coûteux en terme de communication [56]. Feige et Shamir proposent le concept –moins fort– de preuves à témoin indistinguable (*witness indistinguishability*) en 1990 [54]. Ces preuves cachent le témoin utilisé pour prouver une assertion. Intuitivement, dans le cas d'une assertion pour laquelle il existe plusieurs témoins valides ω_is, un protocole est à témoins indistinguables si un vérifieur malhonnête ne peut pas décider quel témoin a été utilisé lors d'une exécution réussie.

Preuve de connaissance zero-knowledge non-interactives Une variantes des preuves ZK consiste à supprimer l'interaction entre le prouveur et le vérifieur ; dans ce cas, on parle de preuves à divulgation nulle de connaissance non interactive, abrégée en NIZK, pour *non-interactive zero-knowledge* en anglais. En 1987, Fiat et Shamir proposent un paradigme pour rendre non-interactive n'importe quelle preuve interactive [47]. De

nombreux schémas de signature dont celles que nous présentons aux Chapitres 5 et 6 en tirent parti.

1.4 Cryptographie basée sur les codes correcteurs

1.4.1 Problèmes difficiles

À l'exception du cryptosystème de McEliece [57] et du schéma de signature CFS [58], la sécurité des schémas cryptographiques basée sur les codes repose principalement sur le problème du décodage par syndrome (*Syndrome Decoding*), prouvé NP-complet en 1978 [57]. Ce point constitue un argument non négligeable quand il s'agit de discuter la sécurité de schémas basés sur la théorie des codes.

Problème du Décodage par syndrome

La version décisionnelle du décodage par syndrome consiste à décider s'il existe un antécédent de petit poids d'un syndrome donné s.

Définition 32. *Soit $H \in \mathcal{M}_{(n-k)\times n}(\mathbb{F}_q)$, $\omega \in \mathbb{N}$ and $s \in \mathbb{F}_q^{n-k}$. La version décisionnelle du problème de décodage par syndrome consiste à décider s'il existe un vecteur $x \in \mathbb{F}_q^n$ vérifiant $H.x^\top = s$ et $\omega t(x) \leqslant \omega$.*

La version calculatoire (*Computational Syndrome Decoding* Problem) consiste à retourner la valeur d'un tel antécédent de petit poids.

Définition 33. *Soit $H \in \mathcal{M}_{(n-k)\times n}(\mathbb{F}_q)$, $\omega \in \mathbb{N}$ and $s \in \mathbb{F}_q^{n-k}$. La version calculatoire du problème de décodage par syndrome consiste à calculer un vecteur $x \in \mathbb{F}_q^n$ vérifiant $H.x^\top = s$ et $\omega t(x) \leqslant \omega$.*

Problème Distance minimale

Le problème de la distance minimale peut se voir comme un cas particulier du décodage par syndrome pour lequel le syndrome s est le syndrome nul.

Définition 34. *Soit $H \in \mathcal{M}_{(n-k)\times n}(\mathbb{F}_q)$, $\omega \in \mathbb{N}$. La version décisionnelle (respectivement calculatoire) du problème de la distance minimale consiste à décider s'il existe (resp. à calculer) un vecteur $x \in \mathbb{F}_q^n$ vérifiant $H.x^\top = 0$ et $\omega t(x) \leqslant \omega$.*

Ce problème a été montré NP-complet plus récemment, par A. Vardy en 1997 [59].

1.4.2 Schémas de Chiffrement

Chiffrement de McEliece

Alors que le cryptosystème RSA [27] fut la première instance remplissant les critères du paradigme à clé publique introduit par Diffie et Hellman [40], McEliece proposa en 1978 un tel système de chiffrement basé sur la théorie des codes [60]. L'idée de base consiste à utiliser comme clé secrète un code C pour lequel il existe un algorithme de décodage efficace. La clé publique associée consiste alors en un code équivalent C' apparaissant aléatoire qui a pour but de masquer la structure du code secret C. C'est la famille des codes de Goppa [17] qui est le plus souvent utilisée pour instancier le cryptosystème de McEliece présenté à la Figure 1.3.

1. Génération des Clés : $\mathsf{KeyGen}(1^\lambda)$
Clé secrète $\mathsf{sk} = (G, S, P)$
G matrice génératrice de $C[n, k, d]$
avec $\mathcal{D}$ un algorithme de décodage.
S une matrice binaire inversible $k \times k$.
P une matrice de permutation $n \times n$.

Clé publique pk
$\mathsf{pk} = (G' = SGP, t = \lfloor \frac{d-1}{2} \rfloor)$

2. Chiffrement : $\mathsf{Enc}(\mathsf{pk}, m)$.
$m \in \{0,1\}^k$.
Retourner $c = mG' + e$
où $e \in \{0,1\}^n$ et tel que $\omega t(e) = t$.

3. Déchiffrement : $\mathsf{Dec}(\mathsf{sk}, c)$.
Puisque $\omega_H(eP^{-1}) = \omega_H(e) = t$,
$mS = \mathcal{D}(cP^{-1})$ *i.e.*,
$mS = \mathcal{D}((mS)G + eP^{-1})$.
Retourner $m = (mS)S^{-1}$.

Figure 1.3: Chiffrement de McEliece

Deux types d'attaques peuvent potentiellement affecter le système de McEliece:

- *Attaque structurelle.* Ce cas correspond au cas où la structure du code C peuvent être retrouver. Le code C' ne peut donc plus être considéré comme aléatoire.
- *Attaque générique.* Dans ce cas, le code public C' et considéré comme aléatoire et ne présentant donc aucune structure apparente.

Une attaque générique sur le système de McEliece reviendrait à attaquer le problème général des codes aléatoires binaires prouvé NP-difficile dans [57], à travers une réduction polynomiale entre ce dernier problème et le probleme du décodage par syndrome. Les codes de Goppa, initialement proposés par McEliece, sont toujours considérés comme une instanciation sure du système McEliece même si certaines attaques [61, 62, 63] ont conduit à reconsidérer les paramètres utilisés [64].

Chiffrement de Niederreiter

En 1986, Niederreiter propose la verison duale du chiffrement de McEliece [65]. Le rôle de la matrice G est désormais joué par la matrice de parité du code H, comme décrit en

Figure 1.4. À paramètres fixés, leurs sécurités sont équivalentes [66]. Pour le système de McEliece, il serait absurde d'utiliser la forme systématique de G' pour chiffrer un message m puisque le chiffré c comporterait la valeur de m en clair sur sa première partie ; cette problématique ne se pose pas dans le cas du système de Niederreiter. En notant H', la matrice de parité H masquée, la mise sous forme systématique permet alors de manipuler des clés publiques de taille divisée par 2.

1. Génération des Clés : KeyGen
Clé secrète $\mathsf{sk} = (G, S, P)$
H matrice de parité de $C[n, k, d]$
avec $\mathcal{D}$ un algorithme de décodage.
S une matrice binaire inversible $(n-k) \times (n-k)$.
P une matrice binaire de permutation $n \times n$.

Clé publique pk
$\mathsf{pk} = (H' = SHP, t = \lfloor \frac{d-1}{2} \rfloor)$

2. Chiffrement : $\mathsf{Enc}(\mathsf{pk}, m)$.
$m \in \{0,1\}^n$ tel que $\omega t(m) = t$.
$c = H'm^\top$.

3. Déchiffrement : $\mathsf{Dec}(\mathsf{sk}, c)$.
Calculer $S^{-1}c = HPm^\top$.
Puisque $\omega t(Pm^\top) \leqslant t$,
$\mathcal{D}$ permet de retrouver $Pm^\top$.
Retourner $m^\top = P^{-1}Pm^\top$.

Figure 1.4: Chiffrement de Niederreiter

Dans la suite, on notera $\mathsf{Enc}^{\text{Nied}}$ et $\mathsf{Dec}^{\text{Nied}}$ les primitives de chiffrement et de déchiffrement relative au cryptosystème de Niederreiter.

1.4.3 Protocole Zero-Knowledge basé sur les Codes

En 1993, Stern propose un protocole d'authentification basé sur les codes rappelé à la Figure1.5. C'est un protocole à trois passes au cours duquel un prouveur va convaincre un vérifieur qu'il connaît la solution à une instance (difficile) du décodage par syndrome. La probabilité de tricher étant de 2/3 à chaque tour, le protocole doit être répété $\lambda/log(3/2)$ fois pour assurer le niveau de sécurité λ.

Données Publiques : $(H, s, \omega) \in \mathcal{M}_{(n-k)\times n}(\mathbb{F}_2) \times {\mathbb{F}_2}^{n-k} \times \mathbb{N}$
$\mathcal{P}$ prouve la connaissance de $z \in \mathbb{F}_2^n$: $Hz^T = s$ et $\omega t(z) = \omega$

1. $\mathcal{P}$ calcule les engagements suivants :
   $\pi \xleftarrow{\$} \Sigma_n$, $u \xleftarrow{\$} \in \mathbb{F}_2^n$
   $c_1 = \mathsf{H}(\pi || Hu^T), c_2 = \mathsf{H}(\pi(u)), c_3 = \mathsf{H}(\pi(z+u))$
2. $\mathcal{P}$ envoie c_1, c_2, c_3
3. $\mathcal{V}$ envoie $\mathsf{ch} \xleftarrow{\$} \{0, 1, 2\}$
4. 3 cas pour $\mathcal{P}$:
   $\mathsf{ch} = 0$: $\mathcal{P}$ envoie π, u
   $\mathsf{ch} = 1$: $\mathcal{P}$ envoie $\pi, z+u$
   $\mathsf{ch} = 2$: $\mathcal{P}$ envoie $\pi(z), \pi(u)$
5. 3 cas pour $\mathcal{V}$:
   $\mathsf{ch} = 0$: $\mathcal{V}$ vérifie c_1, c_2
   $\mathsf{ch} = 1$: $\mathcal{V}$ vérifie c_1, c_3
   $\mathsf{ch} = 2$: $\mathcal{V}$ vérifie c_2, c_3
   et $\omega t(\pi(z)) = \omega$
6. $\mathcal{V}$ retourne *Accept* si les vérifications passent
   $\perp$ sinon.

Figure 1.5: Protocole de Stern

Remarquons que cette version du protocole de Stern présente un cas de distinguabilité.

En effet, en fonction de la valeur du challenge ch, un vérifieur, sera capable de décider si z' a été impliqué durant le protocole :

- cas $\mathsf{ch} = 0$. En plus de vérifier les validités de c_1 et c_2, un tel vérifieur $\mathcal{V}(z')$ peut aussi vérifier si c_3 est égal à $h(\pi(z' + u))$;
- cas $\mathsf{ch} = 1$. En plus de vérifier les validités de c_1 et c_3, un tel vérifieur $\mathcal{V}(z')$ peut aussi vérifier si c_2 est égal à $h(\pi(z + u + z'))$.

Lorsque de telles vérifications additionnelles sont satisfaites, le vérifieur est assuré que sont secret z' est égal au secret z mis en jeu par le prouveur ou au contraire que z' n'est pas égal au secret z le cas échéant. Ainsi, ce protocole n'est pas zero-knowledge. Nous verrons que cette propriété nous conduira à définir et exploiter la notion *Testable weak Zero-Knowledge* au Chapitre 5 où sera présenté un schéma de signature de groupe basée sur les codes. À titre d'information, nous rappelons que Stern a proposé en 1996, une version randomisée de son protocole [67] assurant cette fois la propriété de zero-knwoledge. L'engagement c_i est désormais calculé en concaténant une graine aléatoire r_i aux valeurs initialement données en entrée à H. Par exemple, c_1 est calculé comme suit $c_1 = h(\pi \| Hu^\top \| r_1)$ où r_1 est une graine aléatoire révélée lors de l'étape 3 dans les cas ch $= 0$ et ch $= 1$.

1.4.4 Schémas de signature

Il existe principalement deux approches quand il s'agit de réaliser des schémas de signatures basés sur la théorie des codes. La première repose sur le paradigme "Hacher-et-Signer" ("Hash-and-Sign") qui consiste principalement en la signature CFS [58] et ses variantes [68, 69]. La seconde approche est celle obtenue via le paradigme de Fiat et Shamir [47] qui permet de transformer tout protocole d'authentification à 3 passes en un schéma de signature numérique.

La signature CFS

Le paradigme "Hash-and-Sign" permet de convertir tout schéma de chiffrement à clé publique en un schéma de signature. De manière générique, ce paradigme requiert une fonction à trappe f comme clé publique pour laquelle la clé secrète associée sera la trappe f^{-1} (f peut être vue comme une fonction de chiffrement pour laquelle f^{-1} la fonction de déchiffrement associée). Pour signer un message m, un signataire va d'abord calculer $y = \mathsf{H}(m)$ un haché de m appartenant au domaine d'arrivée de f. La signature sera $\sigma = f^{-1}(y)$ et un vérifieur public acceptera la signature si $f(\sigma) = \mathsf{H}(m)$. Pour assurer la sécurité d'une telle construction, il est nécessaire d'avoir accès à une fonction de hachage dont les sorties sont réparties uniformément sur l'ensemble d'arrivée de f (ou l'ensemble

des chiffrés). Il n'existe pas de telles fonctions dans le cas des codes correcteurs mais en choisissant des codes avec des paramètres adaptés, il est possible de se ramener à une situation pour laquelle la fonction H retourne un chiffré après plusieurs tentatives si nécessaires. C'est le principe de CFS, rappelée à la Figure 1.6, qui repose sur le cryptosystème de Niederreiter instancié avec des codes de Goppa.

1. Paramètres publics : $\mathsf{Setup}(1^\lambda)$
param $= (m, t) \in \mathbb{N}^2$.
h une fonction de hachage.

2. Génération des Clés : $\mathsf{KeyGen}(\text{param})$
Générer une bi-clé Niederreiter $(\mathsf{sk}^{\text{Nied}}, \mathsf{pk}^{\text{Nied}})$ utilisant un Goppa binaire irréductible de paramètres $[2^m, 2^m - mt, 2t + 1]$.

Clé secrète sk
$\mathsf{sk} = \mathsf{sk}^{\text{Nied}}$
Clé publique vk
$\mathsf{vk} = \mathsf{pk}^{\text{Nied}}$

3. Signature : $\mathsf{Sign}(\mathsf{sk}, m)$.
$z = \mathsf{H}(m)$, $i \xleftarrow{\$} \{0,1\}^r$, $s = h(z\|i)$.
Tant que $\mathsf{Dec}^{\text{Nied}}(\mathsf{sk}, s)$ échoue:
$i \xleftarrow{\$} \{0,1\}^r$, avec $r = mt$.
$s = \mathsf{H}(z\|i)$.
$\gamma = \mathsf{Dec}^{\text{Nied}}(\mathsf{sk}, s)$.
Retourner $\sigma = (\gamma, i)$.

4. Vérification : $\mathsf{Verif}(\mathsf{vk}, m, \sigma)$.
$\sigma = (\gamma, i)$.
$s_1 = \mathsf{Enc}^{\text{Nied}}(\mathsf{vk}, m)$.
$s_2 = \mathsf{H}(\mathsf{H}(m)\|i)$.
Si $s_1 = s_2$, Accepter la signature.
Refuser sinon.

Figure 1.6: Signature CFS

Sécurité de CFS et paramètres Il n'est pas possible de considérer la matrice publique CFS comme une fonction à trappe indistinguable d'une matrice aléatoire puisqu'un distingueur a récemment été exhibé [16]. Néanmoins, ce résultat n'a mené à aucune attaque pratique sur le schéma qui reste utilisable. En effet, la sécurité de CFS repose sur la propriété à sens unique du schéma ; les meilleures attaques sont génériques et traitent la matrice publique comme une matrice aléatoire.

Attaquer le schéma CFS signifie être capable de résoudre une instance du décodage par syndrome. Plus précisément, pour un message donné, le but d'un attaquant sera de trouver un vecteur i pour lequel il saura inverser $s = h(h(m)\|i)$: il y a autant d'instances que de choix pour i et il suffit à l'attaquant d'en résoudre une seule; c'est en ce dernier point que le problème diffère de la version calculatoire du décodage par syndrome où l'attaquant n'a pas le choix de l'instance à résoudre. Ce problème peut être attaqué par des algorithmes du type GBA (*General Birthday Algorithm*) [70]. Ainsi, reprenant une attaque sur le schéma CFS due à D. Bleichenbacher, Finiasz et Sendrier [63] proposent les paramètres présentés en Figure 1.7 pour une instanciation sure de CFS.

Le schéma de signature de groupe proposé au Chapitre 5 jouit d'une instanciation s'appuyant le schéma CFS.

	t= 8	t= 9	t= 10	t= 11	t= 12
m=15	$2^{51.0}/2^{51.0}$	$2^{60.2}/2^{43.3}$	$2^{63.1}/2^{55.9}$	$2^{67.2}/2^{67.2}$	$2^{81.5}/2^{54.9}$
m=16	$2^{54.1}/2^{54.1}$	$2^{63.3}/2^{46.5}$	$2^{66.2}/2^{60.0}$	$2^{71.3}/2^{71.3}$	$2^{85.6}/2^{59.0}$
m=17	$2^{57.2}/2^{57.2}$	$2^{66.4}/2^{49.6}$	$2^{69.3}/2^{64.2}$	$2^{75.4}/2^{75.4}$	$2^{89.7}/2^{63.1}$
m=18	$2^{60.3}/2^{60.3}$	$2^{69.5}/2^{52.7}$	$2^{72.4}/2^{68.2}$	$2^{79.5}/2^{79.5}$	$2^{93.7}/2^{67.2}$
m=19	$2^{63.3}/2^{63.3}$	$2^{72.5}/2^{55.7}$	$2^{75.4}/2^{72.3}$	$2^{83.6}/2^{83.6}$	$2^{97.8}/2^{71.3}$
m=20	$2^{66.4}/2^{66.4}$	$2^{75.6}/2^{58.8}$	$2^{78.5}/2^{76.4}$	$2^{87.6}/2^{87.6}$	$2^{101.9}/2^{75.4}$
m=21	$2^{69.5}/2^{69.5}$	$2^{78.7}/2^{61.9}$	$2^{81.5}/2^{80.5}$	$2^{91.7}/2^{91.7}$	$2^{105.9}/2^{79.5}$
m=22	$2^{72.6}/2^{72.6}$	$2^{81.7}/2^{65.0}$	$2^{84.6}/2^{84.6}$	$2^{95.8}/2^{95.8}$	$2^{110.0}/2^{83.6}$

Figure 1.7: Complexités Temps/Mémoire de l'attaque de Bleichenbacher sur CFS.

Heuristique de Fiat-Shamir

Contrairement au paradigme "Hash-and-Sign", la transformation de Fiat-Shamir [47] ne nécessite pas l'utilisation de fonctions à trappe et consiste à transformer un schéma d'identification à 3 passes en un schéma de signature. L'idée est de voir le signataire comme un prouveur alors que le vérifieur sera remplacé par un oracle aléatoire. Considérons un schéma à 3 passes pour lequel le prouveur calcule des engagements regroupés sous la valeur cmt, des défis ch générés par un challengeur et les réponses aux différents challenges contenues dans rsp. L'heuristique de Fiat-Shamir requiert l'utilisation d'un oracle aléatoire H : en lieu et place des défis envoyés par un challengeur, le signataire du message m calcule $\mathsf{ch} = \mathsf{H}(\mathsf{cmt}, m)$. Le secret associé au schéma d'identification, qui constituera la clé de signature, permettra au signataire de générer rsp, constituant les réponses au challenge ch. Ainsi la retranscription $(\mathsf{cmt}, \mathsf{ch}, \mathsf{rsp})$, simulant des interactions entre un prouveur et un vérifieur, constituera la signature σ sur le message m. Une des instanciations les plus célèbres concerne le schéma d'identification de Stern, prouvé Zero-Knowledge, qui peut donc donner lieu à un schéma de signature par le biais de l'heuristique de Fiat-Shamir.

Les avantages du paradigme de Fiat-Shamir repose sur la sécurité prouvée d'un tel protocole dans le modèle de l'oracle aléatoire. Le principal inconvénient est la taille des clés de signatures importantes inhérentes à de nombreuses simulations du protocole interactif.

Le paradigme de Fiat-Shamir permettra la conception des schémas de signature de groupe, présentés aux Chapitres 5 et 6.

1.5 Introduction aux Fuzzy Extractors

Nous verrons au Chapitre 3 que le domaine des *Fuzzy Extractors* nous est apparu comme un réponse adaptée aux problématiques liées à la solution TA. Nous avons choisi de ne pas traduire le terme *Fuzzy Extractor* pour éviter toute modification du sens et conserver

la concision de la terminologie anglaise.

1.5.1 Présentation et Problématique

Le domaine de la réconciliation d'information (*information reconciliation*) permet de retrouver des valeurs identiques à partir de données bruitées et sujettes à d'inhérentes variations. En parallèle, le domaine de la *privacy amplification* [71] permet de convertir des données contenant de l'information en un secret uniformément distribué. Les Fuzzy Extractors (FEs), introduits par Dodis *et al.* [12][4], consistent en un protocole non-interactif (Gen, Rep) réalisant ces deux propriétés.

Plus précisément, un FE est un couple d'algorithmes sécurisés permettant la génération de clés cryptographiques à partir de n'importe quelle information bruitée contenant suffisamment d'entropie. L'idée est d'alors exploiter des sources d'entropie pour les transformer en clés stables qui seront ensuite plongées dans un mécanisme cryptographique plus global. Deux cas d'application sont souvent mentionnés comme possible instanciations des FEs : la biométrie et les PUFs (pour *Physically Unclonable Functions*) introduits par Pappu [72]. En particulier, l'avènement de smartphones adaptés [73, 74, 75, 76, 77] renforce l'intérêt pour l'authentification biométrique qui permettrait de manipuler des données portant une entropie considérable mais délicate à manipuler. En effet, deux signaux biométriques d'une même empreinte –à l'instar d'empreintes digitales, de l'iris ou encore d'une signature manuscrite– sont rarement identiques bien que proches. Les systèmes d'authentification par mot de passe qui consistent généralement pour un serveur à stocker $y = f(w)$ où w est le mot de passe et f une fonction à sens unique ne peuvent donc pas s'étendre à la biométrie. Ainsi, alors que l'accès aux empreintes biométriques se démocratise depuis quelques années, leur protection par les méthodes cryptographiques traditionnelles (chiffrement, signature) n'est donc pas envisageable:

- par conception, la plupart des cryptosystèmes doivent avoir des entrées stables pour assurer leur bon fonctionnement. Par exemple, une différence d'un seul bit entre w et w' entraîne des différences majeures dans les sorties par une fonction à sens unique; de manière analogue, deux chiffrés seront égaux si, a priori, ils sont le résultat du chiffrement du même clair sous la même clé de chiffrement.

- par nature ou par acquisition, deux signaux biométriques –pour une empreinte et un utilisateur donnés– ne seront pas égaux bien qu'identifiant l'utilisateur.

Puisque les empreintes biométriques caractérisent les utilisateurs et contiennent plus d'entropie qu'un simple mot de passe, il apparaît légitime de vouloir en extraire des clés

[4]Dans toute la suite, nous nous référerons à la version longue des mêmes travaux publiés en 2008[13].

cryptographiques. La problématique est donc double : convertir des données bruitées et a priori non uniformes en chaînes de caractères *stables* et *uniformément distribuées*.

Introduits dans cette optique, les FEs se définissent alors à travers deux primitives :

1. **Gen.** Procédure de génération : étant donné un signal bruité w, un FE est capable dans un premier d'en extraire une valeur uniformément distribuée (*i.e.* une clé) R et une donnée de reconstruction publique P. Pour un cas d'usage relatif à l'authentification, cette étape peut se voir comme l'enrôlement d'un utilisateur.

2. **Rep.** Procédure de reproduction : à partir d'une valeur w' et de P, le FE sera capable de reproduire R si w' est *assez* proche de w. Pour un cas d'usage relatif à l'authentification, cette étape peut se voir comme l'authentification d'un utilisateur préalablement enrôlé.

La donnée P doit permettre la tolérance au bruit et sera être stockée publiquement ; il est donc important que R apparaisse aléatoire et ce, même en la présence P. Une des problématiques majeure des FEs est de maximiser la taille de la clé R extraite de w tout en prenant en compte que la publication de P révèle aussi de l'information sur w. En effet, puisque P permet de retrouver R initialement issu de w, c'est que cette donnée contient de l'information sur w. Pour estimer cette perte d'entropie, Dodis *et al.* [13] considèrent dans le modèle de la théorie de l'information que l'entropie "perdue" par la publication de P correspond à sa longueur en bits.

Dans toute la suite, un FE pourra être dénoté génériquement $(\mathsf{Gen}, \mathsf{Rep})$ où Gen et Rep constituent respectivement les procédures de génération et de reproduction.

Dans leur travaux, Dodis *et al.* proposent également le concept de *Secure Sketch* (SS) qui va constituer un brique essentielle dans leurs constructions de FEs. Un tel objet permet de reconstruire un secret bruité w avec une philosophie analogue à celle des FEs: 1) à partir dudit signal w, le SS retourne s tel que 2) à partir d'une version bruitée w' et de s, il sera capable de reproduire w. De même que P ne doit pas révéler d'information à propos de R dans le cas des FEs, ici s ne doit pas révéler trop d'information à propos de w.

Différentes métriques La proximité entre deux signaux w et w' est quantifiée par la notion de métrique. Un espace métrique est un espace $\mathcal{M}$ avec une fonction de distance $d : \mathcal{M} \times \mathcal{M} \to \mathbb{R}^+$ vérifiant les propriétés classiques de symétrie ($\forall x, y,\ d(x, y) = d(y, x)$), séparation ($\forall x, y,\ d(x, y) = 0 \Leftrightarrow x = y$) et l'inégalité triangulaire ($\forall x, y, z,\ d(x, z) \leqslant d(x, y) + d(y, z)$).

Les 3 différentes métriques principalement exploitées dans le cas discret sont les métriques de Hamming, la différence d'ensembles et la distance d'édition.

Hamming Ici $\mathcal{M} = \mathcal{F}^n$ pour un alphabet $\mathcal{F}$. La distance de Hamming $d_H(w, w')$ entre deux mots de même longueur w et w' est le nombre de positions pour lesquels ils diffèrent.

Différence d'ensembles Ici $\mathcal{M}$ consiste en tous les sous-ensembles d'un univers $\mathcal{U}$. Pour deux ensembles w, w', leur différence symétrique est définie par $w \Delta w' \stackrel{def}{=} \{x \in w \cup w' | x \notin w \cap w'\}$. La distance entre deux ensembles w et w', notée d_S, est définie par $d_S(w, w') \stackrel{\text{def}}{=} |w \Delta w'|$.

Notation Un tel espace métrique sera dénoté $(\mathsf{SDif}(\mathcal{U}), d_S)$ tandis que $\mathsf{SDif}_s(\mathcal{U})$ représente tous les ensembles de s éléments issus de l'univers $\mathcal{U}$.

Au cours de leurs travaux, Dodis *et al.* exhibent une équivalence entre une chaîne binaire et un ensemble d'éléments numérotés provenant d'un univers $\mathcal{U}$ de cardinal n. En effet, si w désigne un tel ensemble, il pourra être vu comme le vecteur binaire $x_w \in \{0, 1\}^n$, avec un 1 en position x si $x \in w$, et 0 sinon. Notée $\mathsf{bin\text{-}set}$, cette équivalence n'est alors pas applicable dans le cas où n est superpolynomial en s.

Distance d'édition Ici $\mathcal{M} = \mathcal{F}^*$, et la distance entre w et w' est définie comme le plus petit nombre d'insertions et délétions requises pour transformer w en w'.

Dodis *et al.* ont proposé une méthode générique pour déduire un FE basé sur la distance d'édition à partir d'un FE basé sur la différence d'ensembles.

Dans toute la suite, nous nous intéresserons à la distance de Hamming et à la différence d'ensembles. Comme rappelé ci-dessus, la popularité des FEs est aussi due à leur capacité à répondre élégamment –au moins théoriquement– à des cas d'usage prometteurs et porteurs comme la biométrie ou les PUFs [72]. Tout en soulignant les points d'achoppement liées à de telles applications, nous proposerons les empreintes d'appareils comme nouveau contexte d'application au chapitre 2.7 dont nous discuterons les avantages.

1.5.2 État de l'art

Bien que formalisés en 2004 par Dodis *et al.* [12, 13], certains travaux reprenant la philosophie des Secure Sketchs [78, 79, 80, 15, 81] voire des Fuzzy Extractors [82] avaient déjà été proposés par la communauté. Par l'usage de codes correcteurs, Davida *et al.* [79, 80] ont été les premiers à proposer un protocole d'authentification biométrique *offline*, *i.e.* sans recours une base de données et sans stocker en clair les signaux biométriques des utilisateurs. Ce point permet de s'affranchir de contraintes fonctionnelles (nécessités d'un canal sécurisé et d'une base de données) tout en préservant le respect de

la vie privée des utilisateurs. Bien que novatrices, les constructions proposées reposent sur des contraintes fortes puisqu'une authentification nécessite plusieurs scans de l'empreinte tout en supposant que les erreurs qui s'y produisent soient indépendantes. Juels et Wattenberg répondent à cette problématique en proposant un schéma d'engagement par *décalage de mot de passe* [78]. L'idée de leur construction est d'associer un mot de code c a un secret w pour publier la donnée de reconstruction $P = c + w$. Ainsi, lorsqu'un utilisateur cherche à s'authentifier, il utilise un algorithme de décodage pour retrouver la valeur c à partir de $P - w' = c + w - w'$ pour finalement retrouver $w = P - c$. Alors que bon nombre de Secure Sketchs reprendront cette philosophie [15, 13, 83], des travaux d'optimisation, notamment sur le choix des codes correcteurs utilisés sont également proposés [84, 81]. Puisque naturellement liée aux codes correcteurs, la distance de Hamming était alors sous-jacente à toutes les constructions proposées avant que Juels et Sudan ne permettent la gestion d'ensembles par l'usage d'une voûte floue (*fuzzy vault*) [15]. Leur contribution permet alors de débloquer un secret à condition que les secrets utilisateurs soient proches au sens de la différence d'ensembles. En plus du cas d'usage proposé où deux utilisateurs pourront débloquer un secret en fonction de leurs goûts cinématographiques, cette distance, comme rappelé par les auteurs, apparaît particulièrement adaptée au contexte de la biométrie. Notamment dans le cas des empreintes digitales qui sont souvent représentées sous forme d'ensembles d'éléments caractéristiques (minuties).

En plus d'un important travail de formalisation, Dodis *et al.* proposent de nouvelles constructions de FEs, permettant de gérer les 3 métriques précédemment rappelées et améliorent également les constructions de Secure Sketches sus-citées (taille de la donnée de reconstruction publique P réduite). Exhibant une équivalence entre la représentation ensembliste d'entiers et les vecteurs binaires, Dodis *et al.* ont permis un passage de certaines constructions basées sur la distance de Hamming vers la différence d'ensembles tout en proposant des construction directement adaptées à la différence d'ensembles. Pour ce qui concerne la distance d'édition, les auteurs ont établi une conversion d'une chaîne de caractères w vers un ensemble de sous-chaînes. Cette conversion permet alors de convertir tout FE basé sur la différence d'ensembles vers la distance d'édition.

Il est intéressant de noter que toutes les constructions proposées dans [13] sont basés sur des Secure Sketches, convertis en FEs par une méthodologie générique faisant usage d'extracteurs forts [34] (voir Sous-section 1.5.3).

Boyen [85] a rapidement exhibé un manquement fort quant aux définitions proposées dans [13]. En effet, la sécurité des FEs apparaît implicitement définie dans le cas où un seul appel à la primitive Gen sera réalisé. Plus précisément, tel qu'alors défini, un utilisateur était assuré que la publication de P obtenu via la procédure Gen révélait une quantité d'information contrôlée sur w tout en permettant à l'utilisateur de s'authentifier

-appels à la procédure Rep- autant de fois que souhaité à partir de versions bruitées de w. Cependant, aucune garantie n'était proposée dans le cas où l'utilisateur aurait souhaité enrôler son secret w –via plusieurs appels à la procédure Gen sur w– auprès de différents services, ce qui est précisément le scénario induit par l'usage de la biométrie. En effet, Boyen met en exergue des constructions artificielles de FEs satisfaisant aux définitions introduites par Dodis *et al.* [13] mais facilement exploitables par tout attaquant passif : plusieurs appels à la fonction Gen sur un même secret bruité w produit des données de reconstruction associées $P^1, \ldots, P^\rho$ qui permettront de reconstruire le secret w en totalité. Boyen définit alors les Fuzzy Extractors réutilisables (*Reusable Fuzzy Extractors*) et en propose une instanciation dans un modèle d'erreurs contraignant [5]. La contribution de Boyen peut donc se voir comme une mise en garde quant aux définitions proposées par Dodis *et al.* [13] même si aucune construction de la littérature n'est alors véritablement attaquée.

Par suite, alors que la réutilisabilité sous-entend un modèle d'attaquant passif, Boyen *et al.* proposent en 2005 considère un modèle d'attaquant actif menant au concept de FEs robustes (*Robust Fuzzy Extractors*) [86]. Une telle primitive permet de se prémunir d'une altération de la donnée de reconstruction publique P en la détectant. L'attaquant actif considéré dans ce cas, s'appuie sur le caractère public de P pour en modifier sa valeur avant que l'utilisateur ne fasse appel à la procédure de reproduction Rep. Supposons un utilisateur enrôlé à partir de w tel que $(R, P) \leftarrow \mathsf{Gen}(w)$. Si un attaquant modifie cette valeur P en $\tilde{P}$, l'utilisateur légitime en présentant w' et $\tilde{P}$ lors de la procédure de reproduction de clé va potentiellement reconstruire une clé $\tilde{R} \neq R$. Le concept de FE robuste (*Robust Fuzzy Extractor*) a donc été introduit pour répondre à cette problématique [86] et permet au passage de procéder à une authentification mutuelle. En effet, puisqu'un FE robuste permet de vérifier l'authenticité de P avant de calculer R, cela permettra d'authentifier la provenance de la donnée publique. La travaux proposés dans [86] reposent également sur des Secure Sketchs : en plus de la donnée de reconstruction s classiquement générée par un Secure Sketch, l'idée est d'appliquer un code d'authentification de message ou MAC (*Message Authentication Code*) sur la valeur w. Ainsi, lors de la phase reconstruction, l'utilisateur pourra vérifier la valeur du secret retrouvé grâce au MAC: si le MAC appliqué à w est valide, c'est que la reconstruction de w s'est déroulée comme prévue et que, a priori, la valeur de P n'a pas été modifiée. Ces travaux proposent une méthode générique pour transformer tout FE classique en un FE robuste dans le modèle de l'oracle aléatoire. Les travaux suivants sur les FEs robustes ont permis notamment d'augmenter la tailles des clés extraites tout en proposant des constructions prouvées sures dans le modèle standard [87, 88, 89].

[5] Le modèle proposé requiert que le OU exclusif entre les valeurs prises par le même secret bruité ne révèle pas d'information sur la distribution associée.

En parallèle et faisant suite à la brèche ouverte par Boyen en 2004 [85], plusieurs résultats négatifs voient le jour. D'une part, Simoens *et al.* montrent en 2009 que les SSs basés sur la distance de Hamming, *i.e.* ceux proposés dans [78, 13], ne sont pas réutilisables [90]. D'autre part, Blanton et Aliasgari [91, 92] prolongent ces résultats dans le cas de la différence d'ensembles [15, 13]. Au sortir de ces attaques, il n'existe aucun FE réutilisable, marquant alors une sérieuse limitation à une outil si prometteur théoriquement.

S'attachant à la problématique des tailles de clé et remarquant l'exigence du modèle de la théorie de l'information choisi dans les définitions originelles [13], Fuller *et al.* proposent des FEs pour lesquels la propriété de sécurité repose désormais sur le modèle calculatoire [83]. Tel que définie par Dodis *et al.*, la sécurité d'un FE assure que (R, P) sera statistiquement indistinguable de (U_l, P): Fuller *et al.* proposent donc d'assouplir cette propriété en requérant que cette indistinction se limite à un adversaire limité en puissance de calcul. Fuller *et al.* proposent alors un FE –basé sur le problème LWE [93] et gérant la distance de Hamming– qui permet d'extraire un clé R de longueur égale à l'entropie du secret m. À titre de comparaison, et sans mentionner le cas des FEs robustes, les constructions proposées par Dodis *et al.* [12, 13] ne peuvent pas faire mieux qu'extraire des clés de longueur $m - log(P) - 2log(1/\epsilon)$ bits où, $log(P)$ est généralement proportionnel à la capacité de correction du FE et ϵ quantifie la distance à l'uniforme.

Un point novateur de leur travaux, basé sur la distance de Hamming, est de s'affranchir de la méthodologie classique qui consiste à concevoir un FE à partir d'un SS. Leur idée est de générer une clé R aléatoirement puis de la chiffrer avec le secret w. Finalement, s'appuyant sur LWE, une version bruitée w', à moins de t erreurs, permettra de retrouver R.

Finalement, Canetti *et al.*, proposent le premier FE prouvé réutilisable en 2016 [94]. Cette construction, également basée sur la distance de Hamming, peut se voir comme une extension du schéma de Fuller *et al.* où la clé aléatoirement générée R est cette fois chiffrée par plusieurs sous-chaînes du secret w : ces différents chiffrés constituent alors la donnée de reconstruction P. À paramètres fixés, et avec $w' \approx w$, il sera possible d'extraire au moins une sous-suite en commun entre w et w', permettant alors de déchiffrer R. Puisque faisant usage d'une fonction de hachage, la sécurité de leur construction est assurée dans le modèle de l'oracle aléatoire tandis que leur modèle de sécurité est une adaptation des modèles proposés par Boyen [85].

D'autre part, reprenant l'équivalence exhibée par Dodis *et al.* [13, Section 6], leur construction permet également de gérer la différence d'ensembles dans le cas où les éléments constituant les ensembles sont issus d'un univers de *petite* taille[6]. Cependant

[6]Cela correspond au cas où l'univers devra être de cardinal polynomial en la taille du secret considéré w.

comme rappelé par les auteurs [13] et au Chapitre 4 de ce manuscrit, le cas naturel et de fort intérêt lorsqu'il s'agit de la différence d'ensembles correspond à un univers de *grande* taille où le cardinal de l'univers source des secrets utilisateurs sera superpolynomial en la taille du secret w. Notons qu'avant les travaux présentés au Chapitre 4 de ce manuscrit, il n'existait pas de Fuzzy Extractor, dans ce paradigme du *grand univers*, permettant de gérer la différence d'ensembles qui est particulièrement adaptée à de nombreux contextes (biométrie, *movie lovers'problem* [15], *fingerprinting*, ...).

D'autres Fuzzy Extractors

Les données identifiées comme intéressantes au cours de cette thèse relèvent du cas discret ; il existe cependant des instanciations au cas continu.

FEs dans le cas continu Certaines contributions se sont aussi penchées sur le cas continu comme Linnartz et Tuyls dont les *shieldings functions* [82] peuvent se voir comme des constructions analogues aux FEs définis sur l'espace continu $\mathbb{R}^n$. D'autres constructions traitant du cas continu et adaptées à la biométrie [95, 96, 97, 98, 99] ont été proposées dont certaines mériteraient des considération de sécurité plus abouties []. Nous invitons le lecteur à s'y référer pour plus de détails.

Reverse FEs Les PUFs [72] sont des systèmes physiques qui, lorsque stimulés, génèrent une réponse. Si le même stimulus est donné au PUF plusieurs fois, les réponses seront proches mais généralement pas identiques d'où l'intérêt des FEs dans ce contexte. La sortie du PUF peut alors se voir comme le secret utilisateur w. Les tokens RFID basés sur des PUFs sont très utilisés dans le contexte du paiement électronique mais sont limités en temps de calcul : les algorithmes de reconstruction ou de décodage sous-jacents aux FEs ne pourront pas s'y dérouler. van Herrewege *et al.* ont alors proposé le concept de *Reverse Fuzzy Extractors* [100] qui consiste à ne faire effecteur la primitive Rep que par le vérifieur RFID de puissance de calcul supérieure. Ainsi, à chaque phase d'authentification, le PUF génère un nouveau secret et une nouvelle donnée de reconstruction. C'est alors au vérifieur de s'assurer qu'il est effectivement capable, à partir de cette nouvelle donnée de reconstruction et du secret initial de reproduire le nouveau secret généré par le PUF. Ce scénario permet de plus, d'assurer une authentification mutuelle des intervenants.

1.5.3 Définitions formelles

Sauf mention contraire, les définitions présentées ici sont tirées de [13].

Min-Entropie, Distance Statistique, Hachage Universel et Extracteurs Forts

Quand il s'agit d'estimer la sécurité, on s'intéresse souvent à la probabilité qu'un adversaire prédise une valeur aléatoire (*e.g.* une clé secrète). La meilleure stratégie qu'un adversaire puisse adopter est de deviner la valeur la plus probable de se produire. On définit la *prédictibilité* (*predictability*) d'une variable aléatoire X comme étant $\max_x \Pr[X = x]$. À cette valeur, il est commun de faire correspondre la *min-entropie* (*min-entropy*) $H_\infty(X)$, définie par $H_\infty(X) = -\log(\max_x \Pr[X = x])$.

La min-entropie d'une distribution indique le nombre de bits presque uniformément distribués que l'on pourra en extraire. Le terme "presque" est formellement définie par la notion de *distance statistique* définie comme suit. La distance statistique entre deux distributions X et Y est $\mathbf{SD}(X, Y) = \frac{1}{2}\Sigma_v|\Pr(X = v) - \Pr(Y = v)|$.

Puisqu'un attaquant peut avoir obtenu, par ailleurs de l'information liée à la distribution d'où sont extraits des secrets, il est important de définir la min-entropie d'une distribution sachant une information possiblement corrélée. Dodis *et al.* ont donc également défini la min-entropie moyenne ou *average min-entropy* comme suit.

Définition 35 (Min-entropie moyenne)**.** *Soient X, Y deux distributions de probabilités. La min-entropie de X étant donnée Y est: $\tilde{H}_\infty(X|Y) \stackrel{def}{=} -log(\mathbb{E}_{y\leftarrow Y}[max_x Pr[X = x|Y = y]) = -log(\mathbb{E}_{y\leftarrow Y}[2^{-H_\infty(X|Y=y)}])$.*

Les extracteurs forts (Définition 26) peuvent extraire au plus $l = m - 2\log(\frac{1}{\epsilon}) + O(1)$ bits. Nous définissons désormais les fonctions de hachage universelles qui constituent des extracteurs forts particuliers.

Définition 36 (Fonction de hachage universelles)**.** *On dit que la famille de fonctions de hachage $\{H_x : \{0,1\}^n \to \{0,1\}^l\}_{x\in X}$ est universelle si : $\forall a \neq b \in \{0,1\}^n, Pr_{x\in X}[H_x(a) = H_x(b)] = 2^{-l}$.*

Il est rappelé dans [13] que les fonctions de hachage universelles sont des extracteurs forts à condition que $l \leqslant m - 2\log(\frac{1}{\epsilon}) + 2$.

De manière analogue à la min-entropie moyenne, l'incertitude sur une variable aléatoire peut être conditionnée par la connaissance d'une information auxiliaire. Ainsi, la notion d'extracteur fort est généralisée en extracteur fort dans le cas moyen.

Définition 37 (Extracteur fort dans le cas moyen)**.** *Soit $\mathsf{Ext} : \{0,1\}^n \to \{0,1\}^l$ une fonction probabiliste en temps polynomial qui utilise r bits d'aléa. On dit que Ext est un (n, m, l, ϵ)-extracteur fort dans le cas moyen si pour toute paire de distributions (W, I) telle que $\tilde{H}_\infty(W|I) \geqslant m$, on a $\mathbf{SD}((\mathsf{Ext}(W; X), X, I), (U_l, X, I)) \leqslant \epsilon$, où X est uniforme sur $\{0,1\}^r$.*

Tout extracteur fort peut se convertir en un extracteur fort dans le cas moyen au prix d'une concession sur les paramètres [13]. Les rappels que nous venons d'effectuer nous permettent de définir formellement Secure Sketches et FEs.

Secure Sketch

Une Secure Sketch permet de reconstruire un secret w à partir de w' et d'une donnée publique générée au préablable.

Définition 38. *Un $(\mathcal{M}, m, \tilde{m}, t)$-Secure Sketch est une paire de procédure randomisées, "Sketch" (SS) et "Recover" (Rec) vérifiant les propriétés suivantes:*

1. *La procédure SS prend en entrée $w \in \mathcal{M}$ et retourne $s \in \{0,1\}^*$;*
2. *La procédure Rec prend en entrées un élément $w' \in \mathcal{M}$ et $s \in \{0,1\}^*$. La propriété de consistance (correctness) garantit que si $d(w, w') \leqslant t$, alors $\mathsf{Rec}(w', \mathsf{SS}(w)) = w$. Si $d(w, w') > t$, aucune garantie n'est assurée sur la sortie Rec;*
3. *La propriété de sécurité garantit que pour toute distribution W sur $\mathcal{M}$ de min-entropie m, la valeur de W peut être retrouvée par l'adversaire qui observe s avec une probabilité inférieure ou égale à $2^{-\tilde{m}}$. Autrement dit, $\tilde{H}_\infty(W|\mathsf{SS}(W)) \geqslant \tilde{m}$.*

Un Secure Sketch est efficace si SS et Rec terminent en temps polynomial.

Perte d'entropie La quantité $\tilde{m}$ est appelée entropie résiduelle et la quantité $m - \tilde{m}$ est appelée perte d'entropie (*entropy loss*) d'un Secure Sketch. Les analyses de sécurité proposées par Dodis *et al.* consistent à minorer l'entropie résiduelle, ce qui revient à majorer la perte d'entropie.

Fuzzy Extractors

Nous nous intéressons désormais aux Fuzzy Extractors.

Définition 39. *Un $(\mathcal{M}, m, l, t, \epsilon)$-fuzzy extractor est une paire de procédures randomisées, "Generate" (Gen) et "Reproduce" (Rep) vérifiant les propriétés suivantes:*

1. *La procédure Gen prend en entrée $w \in \mathcal{M}$ et retourne la chaîne binaire extraite $R \in \{0,1\}^*$ et une aide à la reconstruction $P \in \{0,1\}^*$;*
2. *La procédure Rep prend en entrées $w' \in \mathcal{M}$ et une chaîne binaire $P \in \{0,1\}^*$. La propriété de consistance garantit que si $d(w, w') \leqslant t$ et R, P ont été générés par $(R, P) \leftarrow \mathsf{Gen}(w)$, alors $\mathsf{Rep}(w', P) = R$. Si $d(w, w') > t$, aucune garantie n'est assurée sur la sortie Rep ;*

3. *La propriété de sécurité garantit que pour toute distribution W sur $\mathcal{M}$, la chaîne r est presque uniforme même pour ceux qui observent P: si $(R, P) \leftarrow \mathsf{Gen}(W)$, alors $\boldsymbol{SD}((R, P), (U_l, P)) \leqslant \epsilon$.*

Un Fuzzy Extractor est dit efficace si Gen *et* Rep *terminent en temps polynomial.*

Dit autrement, un FE permet d'extraire un aléa R à partir de w et de reproduire R avec succès à partir toute valeur w' assez proche de w. Cette reproduction s'appuie alors sur l'aide à la reconstruction P produite lors de la génération: cette valeur est publique et d'après la propriété de sécurité, R apparaît aléatoire étant donné P. On a les définitions suivantes :

- **Perte d'entropie.** On définit $m - l$ comme étant la *perte d'entropie* d'un FE ;
- **FE cas moyen.** Un FE dans le cas moyen (*average case Fuzzy Extractor*) est un FE pour lequel si $\tilde{H}_\infty(W|I) \geqslant m$, alors $\mathbf{SD}((R, P, I), (U_l, P, I)) \leqslant \epsilon$ pour toute variable aléatoire auxiliaire I.

Fuzzy Extractor à partir d'un Secure Sketch Un Secure Sketch permet de retrouver w à partir d'une version bruitée w' alors qu'un extracteur fort permet d'extraire l'entropie contenue dans w, il apparaît relativement naturel de construire un FE à partir de l'utilisation conjointe d'un Secure Sketch et d'un extracteur. Plus formellement, on a le lemme suivant.

Lemme 2. *Soient* (SS, Rec) *un* $(\mathcal{M}, m, \tilde{m}, t)$*-Secure Sketch et* Ext *un* $(n, \tilde{m}, l, \epsilon)$*-extracteur fort dans le cas moyen. Alors le couple* (Gen, Rep) *tel que décrit ci-dessous est un* $(\mathcal{M}, m, l, t, \epsilon)$*-Fuzzy Extractor :*

- $\mathsf{Gen}(w; r, x)$: *Fixer* $P = (\mathsf{SS}(w; r), x)$, $R = \mathsf{Ext}(w; x)$, *et retourner* (R, P) ;
- $\mathsf{Rep}(w', (s, x))$: *Retrouver* $w = \mathsf{Rec}(w', s)$ *et retourner* $R = \mathsf{Ext}(w; x)$.

Sécurité calculatoire Alors que toutes les constructions proposées par Dodis *et al.* suivent cette méthodologie et voient leur sécurité définie d'un point de vue la théorie de l'information, Fuller *et al.* [83] proposent de satisfaire à une définition basée sur le modèle calculatoire, moins exigeant, dans l'espoir de minimiser la perte d'entropie. Dans ce cas, l'idée est de générer une clé R calculatoirement indistinguable de l'aléatoire plutôt que statistiquement.

Étudier, d'un point de vue calculatoire, la distance entre deux distributions revient à quantifier la proximité de ces distributions aux yeux d'un programme arbitraire dénommé *distingueur* et noté D. On ne fait généralement que peu de suppositions sur un distingueur

si ce n'est qu'il doit retourner $b \in \{0,1\}$. Ainsi, des distributions calculatoirement proches sont des distributions pour lesquelles il n'existe pas de distingueur efficace dont les sorties diffèrent sensiblement en fonction de la distribution étudiée.

Définition 40. *Soient X et Y deux distributions de probabilités. On définit la distance calculatoire entre X et Y comme étant :*

$$\delta^D(X,Y) = |\mathbb{E}[D(X)] - \mathbb{E}[D(Y)]|.$$

On étend cette définition à toute classe de distingueur $\mathcal{D}$ en prenant le maximum sur l'ensemble des $D \in \mathcal{D}$.

Nous rappelons alors la définition calculatoire des FEs telle que proposée par Fuller *et al.* à une différence près : nous ne considérons pas le cas où le FE échoue dans sa reconstruction avec une probabilité δ comme introduit par Dodis *et al.* [13]. Nous avons alors la définition suivante.

Définition 41 (Fuller*et al.*[83])**.** *Soit $\mathcal{W}$ une famille de distribution de probabilités sur l'espace métrique $\mathcal{M}$. Une paire de procédures randomisées "generate" (Gen) et "reproduce" (Rep) est un $(\mathcal{M}, \mathcal{W}, l, t)$-Fuzzy Extractor calculatoire de sécurité $(\epsilon_{sec}, s_{sec})$ si Gen et Rep satisfont les propriétés suivantes :*

- *La procédure de génération Gen prend en entrée $w \in \mathcal{M}$ et retourne une clé extraite $R \in \{0,1\}^l$ et une chaîne de reconstruction $P \in \{0,1\}^*$*
- *La procédure de reconstruction Rep prend en entrées $w' \in \mathcal{M}$ et une chaîne binaire $P \in \{0,1\}^*$. La propriété de consistance garantit que si $d(w,w') \leqslant t$ et $(R,P) \leftarrow$ Gen(w), alors Rep$(w',P) = R$. Si $d(w,w') > t$, alors aucune garantie n'est fournie sur la sortie de Rep.*
- *La propriété de sécurité garantit que pour toute distribution $W \in \mathcal{W}$, la valeur R est pseudo-aléatoire sachant P. Autrement dit, si $(R,P) \leftarrow$ Gen(W) alors, on a $\delta^{\mathcal{D}_{sec}}((R,P),(U_l,P)) \leqslant \epsilon_{sec}$.*

La restriction à une famille de distributions $\mathcal{W}$ est une façon plus générique de spécifier la min-entropie (Définition 39), nous jonglerons entre ces deux considérations au Chapitre 4. D'autre part, comme précisé par Fuller *et al.*, tout FE respectant la définition 39 sera en particulier un Fuzzy Extractor calculatoire tel que défini ci-dessus.

Toujours dans [83], Fuller *et al.* ont proposé un Fuzzy Extractor (calculatoire), non basé sur un Secure Sketch, permettant ainsi d'extraire un clé de taille égale à l'entropie du secret bruité. Leur schéma peut se voir comme une adaptation de la construction

code-offset sur $\mathbb{F}_q$. Le secret R est généré aléatoirement puis transmis via P en tant que mot de code bruité, où cette fois-ci le bruit est joué par w. Ainsi, la connaissance de $w' \approx w$ permettra d'enlever le bruit pour retrouver R. La sécurité de leur schéma repose sur le décodage linéaire d'un code, prouvé NP-difficile [57] et sur le problème LWE [93].

Reusable Fuzzy Extractors Comme nous l'avons vu au cours de l'état de l'art, Canetti *et al.* ont proposé le premier FE –selon la Définition 41– prouvé réutilisable. Basée sur la distance de Hamming, leur construction jouit d'une sécurité prouvée dans le ROM et propose un modèle de sécurité qui pourra se voir comme intermédiaire à ceux proposés Boyen [85]. Encore une fois, nous nous restreignons au cas le FE n'échoue pas à reconstruire R si w' est proche de w.

Définition 42 (Canetti *et al.* [94])**.** *Soit $\mathcal{W}$ une famille de distributions de probabilité sur $\mathcal{M}$. Soit (Gen, Rep) un $(\mathcal{M}, \mathcal{W}, l, t)$-FE calculatoire de sécurité $(\epsilon_{sec}, s_{sec})$. Soient $W^1, W^2, \ldots, W^\rho$ ρ variables corrélées telles que pour tout j, $W^j \in \mathcal{W}$. Soient un adversaire D et le jeu suivant pour tout $j = 1, \ldots, \rho$:*

- ***Échantillonnage*** *Le challengeur tire $w^j \leftarrow W^j$ and $u \in \{0,1\}^l$.*
- ***Génération*** *Le challenger calcule $(R^i, P^i) \leftarrow \mathsf{Gen}(w^i)$.*
- ***Distinction*** *L'avantage de D est :*

  $Adv(D) \stackrel{def}{=} Pr[D(R^1, \ldots R^{j-1}, R^j, R^{j+1}, \ldots, R^\rho, P^1, \ldots, P^\rho) = 1] - Pr[D(R^1, \ldots R^{j-1}, \mu, R^{j+1}, \ldots, R^\rho, P^1, \ldots, P^\rho) = 1]$

(Gen, Rep) est $(\rho, \epsilon_{sec}, s_{sec})$-réutilisable si pour tout $D \in \mathcal{D}_{s_{sec}}$ et pour tout $j = 1, \ldots, \rho$, l'avantage de D est au plus ϵ_{sec}.

Comparaison avec le modèle de Boyen Boyen a proposé deux modèles de sécurité pour définir les FEs réutilisables [85]. Le premier, appelé "outsider security" [85], permet à l'adversaire d'accéder aux valeurs $P^1, \ldots, P^\rho$ qui essaie donc de retrouver un secret w^j ou une clé R^j valide.

Le deuxième modèle, appelé "insider security", confère à l'adversaire la possibilité de contrôler certains serveurs d'authentification. Autrement dit, l'adversaire est alors capable d'appeler la procédure de reproduction Rep sur n'importe quelle donnée de reconstruction valide P^j ou volontairement modifiée $\tilde{P}^j$. Lorsque l'adversaire choisit de ne pas modifier P^j, il apprend alors la valeur de la clé associée R^j comme c'est le cas du modèle de sécurité tout juste défini (Définition 42). D'autre part, cette possibilité d'appeler Rep sur une donnée de reconstruction modifiée vise à traiter le cas des attaques actives liées aux FEs robustes. Une autre différence concerne les secrets considérés : alors que Boyen [85]

utilise des fonctions de perturbation permettant d'obtenir des versions bruités d'un secret w, Canetti *et al.* considèrent simplement des secrets provenant de distributions corrélées.

Ainsi, le modèle proposé par Canetti *et al.* se trouve à mi-chemin entre les définitions de Boyen pusiqu'ils ne s'intéressent pas au cas des FEs robustes. Sur ce point, rappelons que réutilisabilité et authentification mutuelle peuvent se traiter séparément d'autant plus que les travaux de Boyen *et al.* proposent une méthodologie, dans le ROM, pour rendre robuste tout FE [86].

Chapitre 2 :

Stade TA1 - Présentation

Contents

En présentant le projet TA, le but de ce chapitre est de poser le cadre des travaux présentés dans ce manuscrit. L'idée sera de donner une vue d'ensemble du projet TA en orientant les débats vers les problématiques qui nous sont propres. Le stade TA1 présentera l'état de la solution proposé par equensWorldline en réponse à l'appel à projet de la Société Générale. La fin de ce chapitre proposera alors un récapitulatif des requis que la solution TA devra respecter tout en soulignant les limitations du stade TA1 au regard de la vie privée des utilisateurs finaux. Toutes les notions et définitions présentées dans ce chapitre sont informelles et ont un but illustratif ; une formalisation sera proposée dans le prochain chapitre.

2.1 Contexte industriel

Worldine (equensWorldline depuis 2016) est un spécialiste du domaine du paiement dans un contexte B2B2C (pour *Business to Business to Customer*). Plus concrètement, equensWorldline fournit au banques des solutions de paiement, qui elles-mêmes les adaptent alors à leurs clients qui constituent les utilisateurs finaux. Deux cas d'usage majeurs sont identifiés dans l'activité de equensWorldline :

- le paiement de proximité où l'utilisateur procède à son achat chez le commerçant via, en général, une lecture de sa carte bancaire par un terminal de paiement électronique (TPE) ;
- le paiement en ligne où l'utilisateur, à distance, procède à un paiement via un appareil connecté (ordinateur, smartphone, tablette) transmettant des données liée à sa carte de paiement.

Tendances du marché

Le paiement en ligne est désormais installé dans les habitudes des utilisateurs. Faire transiter, à chaque transaction, des données bancaires sur le net est risqué pour les utilisateurs. Les mal-nommés porte-monnaies électroniques agissent comme des tiers permettant d'authentifier les utilisateurs en leur attribuant un couple (identifiant, mot de passe) moins sensibles à manipuler que lesdites données bancaires. *PayPal*, *PayLib* ou encore *V.me* et *Masterpass* respectivement portées par les groupements Visa et Mastercard sont de telles solutions ; *Paylib* est une solution fournie par equensWorldline à de nombreuses banques telles que la BNP Paribas, la Société Générale ou encore la Banque Postale.

Le paiement de proximité a récemment connu un renouveau ces dernières avec l'émergence du paiement sans contact. Que ce soit la carte bancaire de l'utilisateur ou son smartphone, ce scénario de paiement permet une expérience utilisateur améliorée par l'usage de la technologie Near Field Communication (NFC). Dans le cas du paiement mobile, la communication avec le TPE se fait par l'intermédiaire d'un élément sécurisé [101] présent sur le téléphone. Ce rôle est souvent joué par la carte SIM (*Subscriber Identity Module*) de sorte que les fournisseurs de telles solutions doivent s'entendre avec les opérateurs téléphoniques. La technologie Host Card Emulation (HCE) permet de s'affranchir de la présence d'un élément sécurisé [102] et propose donc une alternative logicielle vers laquelle les banques se tournent. Comme nous le verrons, la contrainte principale éditée par la Société Générale, avant l'avènement de HCE était également de fournir une solution ne reposant pas sur un élément sécurisé.

Pour assurer interopérabilité et niveau de sécurité adéquat, de telles solutions de paiement doivent respecter certains standard et recommandations éditées par des autorités reconnues.

Autorités dans le monde du paiement

En plus des bonnes pratiques éditées par les états, les sociétés de carte de paiement publient des spécifications que toute solution devra respecter pour être certifiée.

Sociétés de cartes de paiement Pour equensWorldline, les groupements EMV (pour Europay Mastercard Visa) et CB (pour Cartes Bancaires) [103] constituent les principales autorités à convaincre de la bonne qualité des solutions proposées. Il existe certains standards à respecter :

- La norme PCI DSS (*Payment Card Industry Data Security Standard*) a été élaborée par les sociétés de carte de paiement, c'est-à-dire American Express, Discover Financial Services, JCB International, MasterCard Worldwide et Visa Inc, pour aider à faciliter une large adoption de mesures de sécurité des bases de données. Ces mesures sont d'ordre structurel (contrôle d'accès, réseau sécurisé; ...) et techniques (sécurité de l'information, protection de données sensibles,...).

- En vigueur depuis 2008 en France, le protocole 3-D Secure, portés par Visa et Mastercard, est une solution d'authentification en ligne. Avant de terminer son paiement, l'utilisateur reçoit un SMS sur son téléphone et le rentre sur son navigateur pour réaliser une authentification forte en prouvant la possession dudit téléphone. Avec le dispositif 3-D Secure, ce sont désormais les banques qui sont responsables transactions frauduleuses et des éventuels dédommagements consécutifs.

- L'usage de la technologie HCE, qui permet à un smartphone d'émuler un carte bancaire, à des fins de paiement est également sujet à des certifications Visa [104] et Mastercard [105]. Obtenir ces certifications, à renouveler tous les ans, est un objectif majeur de equensWorldline.

Via l'appel à projet de la SG, le paiement 3-D Secure a indirectement conduit au projet TA tandis que l'émergence de la technologie HCE a apporté de nouveaux cas d'usage dont certaines problématiques reprennent celles de TA (*e.g.* solution exclusivement logicielle).

Autorités étatiques Le paiement fait nécessairement appel à des notions d'authentification et donc de cryptographie. equensWorldline, acteur majeur dans les solutions de paiement doit se plier à des règles et bonnes pratiques (*e.g.* gestion de clés cryptographiques) concernant les systèmes d'information qui, en France, sont

principalement portées par l'Agence Nationale de la Sécurité des Systèmes d'Information (ANSSI) [106, 107]. Aux États-Unis, le NIST [108] (*National Institute of Standards and Technology*) joue le même rôle et a une portée plus internationale.

D'autre part, la nature des informations potentiellement manipulées par equensWorldline pose la question du respect de la vie privée des utilisateurs. La Commission Nationale de l'Informatique et des Libertés (CNIL) a alors pour rôle d'éditer les requis à satisfaire pour tout système d'information [109].

2.2 Les contours du projet Trusted Authentication (TA)

Dans un premier temps, nous résumons l'appel à projet formulé par la SG avant de rappeler certaines notions ou définitions inhérentes aux solutions d'authentification selon equensWorldline. Finalement, nous établirons de nouvelles définitions propres à la philosophie du projet TA.

2.2.1 Description de l'appel à projet de la SG

Le projet TA est la réponse à un appel à projet lancé par la Société Générale (SG) en 2008, client historique d'equensWorldline : l'objectif est de fournir une solution *exclusivement* logicielle d'authentification forte d'un utilisateur $\mathcal{U}$ sur son smartphone. Par extension, nous considérons authentification de $\mathcal{U}$ à travers $\mathcal{T}$, l'ensemble de ses terminaux possiblement réduit à un smartphone seul. Le terme *exclusivement* logiciel sous-entend que la sécurité du paiement par téléphone ne pourra pas reposer sur un élément (matériel) sécurisé (de type carte SIM) [101]. Puisque l'utilisation d'une carte SIM en tant qu'élément sécurisé sur le smartphone sous-entend la coopération de l'opérateur téléphonique concerné, elle implique un certain prix dont les banques cherchent à s'affranchir. En ce sens, le projet TA a alors permis d'anticiper l'émergence de la technologie HCE rappelée ci-dessus.

En plus des pratiques induites par le contexte industriel rappelé en Section 2.1, le produit TA répondra principalement aux objectifs et contraintes de l'appel à projet dont un résumé pourrait être :

- **OP.** l'objectif principal (OP) est de réaliser une solution d'authentification forte ;
- **C1.** La première contrainte (C1) est de proposer une solution n'impliquant aucune tierce partie donc en particulier permettant de s'affranchir de l'usage de la SIM. La solution sera donc potentiellement exécutée en milieu hostile ; la solution proposée ne bénéficiera donc d'aucun moyen de stockage sécurisé.

Clients de TA La réponse d'equensWorldline, décrite dans la suite de cette section, a été vendue à de nombreux clients dont une liste non exhaustive pourrait être la suivante : Société Générale, BNP Paribas, Crédit Agricole, Crédit du Nord, Bancontact Mistercash (BCMC), BforBank, AXA Banque, Postbank, Visa.

Dans toute la suite, nous nous référons au cas d'usage décrit ci-dessus où une application logicielle App permet l'authentification forte d'un utilisateur $\mathcal{U}$ muni de son smartphone -voire de l'ensemble de ses terminaux- dénoté(s) $\mathcal{T}$.

2.2.2 Authentification forte

Dans cette sous-section, nous définissons certains concepts sur lesquels nous allons nous appuyer tout au long de ce manuscrit. Nous commençons par une définition de l'authentification forte pour ensuite définir de manière générique un protocole d'authentification selon equensWorldline par la Figure 2.1.

Bien qu'il ne soit pas aisé de trouver une définition arrêtée et formelle de l'authentification forte, il est communément admis qu'une session d'authentification *forte* est une session qui s'appuie sur plusieurs facteurs d'authentification que l'ANSSI semble définir selon quatre catégories [107]: ce qu l'on sait, ce que l'on a, ce que l'on est et ce que l'on sait faire. Un facteur d'authentification permet de classifier une information fournie par un utilisateur pour s'authentifier (*e.g.* le mot de passe est un facteur de connaissance). Nous proposons alors la définition suivante.

Définition 43 (Authentification forte)**.** *On appelle une authentification forte une session d'authentification au cours de laquelle l'utilisateur s'authentifie en prouvant la validité d'au moins, deux facteurs d'authentification (FA) parmi les suivants :*

FA-1 *Ce que l'on sait (e.g. un mot de passe) ;*

FA-2 *Ce que l'on a (e.g. une carte à puce) ;*

FA-3 *Ce que l'on est (e.g. une empreinte digitale), ou autrement dit biométrie physiologique;*

FA-4 *Ce que l'on fait/sait faire/peut faire (e.g. une signature manuscrite), ou autrement dit le comportement utilisateur.*

Nous définissons volontairement le facteur FA-4 de manière plus globale que l'ANSSI afin de ne pas l'assimiler à de la biométrie comportementale pure. En effet, nous verrons que le comportement de l'utilisateur transparaîtra à travers l'usage qu'il fera de ses terminaux $\mathcal{T}$, de type smartphone. Usuellement restreint à la biométrie comportementale classique (*e.g.* signature, démarche, voix), nous étendrons alors le facteur FA-4 à certaines

valeurs caractéristiques d'un comportement sur $\mathcal{T}$ (*e.g.* liste des application installées, préférences utilisateurs). Ces données sont de plus en plus étudiées pour identifier et traquer les utilisateurs sur l'internet via les domaines des empreintes de navigateurs et d'appareils (*browser fingerprinting* [110] et *device fingerprinting* [111]).

Exemple 1. *Une méthode d'authentification forte très répandue à l'heure actuelle est celle le résultat de la norme 3-D Secure Entrer sur son navigateur web l'OTP (One Time Password) reçu sur son téléphone -non nécessairement un smartphone- assure la possession dudit téléphone (FA-2) tandis que la possibilité de l'utiliser assure la connaissance du PIN associé (FA-1).*

2.2.3 Réponse d'equensWorldline à l'appel à projet

La solution proposée par equensWorldline consiste en une session d'authentification telle que décrite à la Figure 2.1.

Protocole générique d'authentification

Nous considérons ici le cas d'une authentification unidirectionnelle au cours de laquelle un serveur equensWorldline vérifie l'identité d'un utilisateur. On suppose que l'utilisateur s'adresse effectivement au serveur par une vérification classique de certificats cryptographiques. Alors que différentes méthodes d'authentification (Proof of Knowlegde, Signature) ont été explicitées dans le premier chapitre (partie 1.3.3), nous présentons le fonctionnement générique d'un protocole d'authentification dans le paradigme TA, défini par le triplet d'algorithmes (authGen, authProve, authVerif).

1. authGen(sk) génère un clé publique pk associée au secret sk préalablement enrôlé.
2. authProve(sk, pk; μ) génère π une preuve que pk est issue de sk.
3. authVerif(π, pk) retourne 1 si π a bien été générée à partir du secret sk associée à pk; 0 sinon.
4. La propriété de sécurité repose sur deux points :
   - retrouver le secret sk à partir des données publiques (dont pk) est un problème difficile.
   - les preuves générées π_is ne sont pas rejouables ni utilisables pour s'authentifier au nom de l'utilisateur légitime.

Figure 2.1: Description informelle d'une authentification TA

Afin de prouver son identité, un utilisateur devra donc prouver la connaissance d'un secret sk qui caractérise son identité publique PK, qui aura été préalablement enrôlée. En

fonction du contexte, une authentification TA pourra être mise en œuvre par l'utilisation de Proof of Knowledge ou encore de signature numérique. Dans le contexte TA, aucun moyen de stockage sécurisé n'est mis à disposition de l'utilisateur ; ce sont donc la génération et la gestion du secret sk qui seront les étapes critiques au regard de la contrainte C1. D'autre part, afin d'assurer une authentification forte, la génération dudit secret sk devra également faire la preuve de l'implication de différents facteurs d'authentification.

Nouveaux objectifs et nouvelles contraintes

Du fait de la contrainte C1 relativement forte, equensWorldline a proposé une solution s'appuyant sur le plus de facteurs de confiance possible. La philosophie de la solution est donc la suivante : l'utilisateur $\mathcal{U}$ devra être capable de prouver simultanément qu'il est le bon utilisateur $\mathcal{U}$ utilisant le bon smartphone (resp. ensemble de terminaux) $\mathcal{T}$ par usage de la bonne application logicielle App. Reprenant la Figure 2.1, l'utilisateur devra prouver la connaissance d'un secret qui est propre à son identité publique et assurant ces différents objectifs. Worldline a donc proposé la solution *Trusted Authentication* (TA) pour laquelle une session d'authentification devra satisfaire la définition suivante.

Définition 44. *En considérant un utilisateur $\mathcal{U}$ muni des terminaux $\mathcal{T}$ où est installée l'application logicielle App, une authentification TA est une session d'authentification telle que définie par la Figure 2.1 répondant aux objectifs suivants :*

O1 Authentifier l'utilisateur.

O2 Authentifier ses terminaux.

O3 Authentifier son application logicielle App.

O4 Prouver la connaissance d'un secret valide assurant que O1, O2 et O3 sont conjointement remplis.

Une authentification TA est une session d'authentification satisfaisant O4.

Remarque 1. *Une authentification TA est nécessairement une authentification forte.*

Par la contrainte C1, $\mathcal{T}$ ne dispose pas d'élément sécurisé ; la manipulation de données sensibles devra alors être minimisée. En particulier, il sera dangereux d'y stocker sk pour de futures sessions d'authentification. Ce secret devra alors être (re-)généré quand nécessaire c'est-à-dire lors des phases d'enrôlement et d'authentification.

Contrainte de la vie privée En plus de la contrainte C1 établie par la SG, l'État, la CNIL notamment [109] impose le respect de la vie privée des utilisateurs. D'autre part, les clients et le grand public y sont de plus en plus sensibilisés. Nous définissons alors la contrainte C2 qui sera traitée en filigrane tout au long de ce manuscrit :

C2. Vie privée. equensWorldline ne doit pas pouvoir manipuler des données utilisateurs considérées comme privées. Seule l'application App déployée en local sur $\mathcal{T}$ en est capable. Ainsi, l'application App doit assurer un service personnalisé sans qu'equensWorldline n'apprenne rien de cette personnalisation.

Mise en perspective A travers les exemples 2 et 3, nous exhibons ce que la solution TA ne pourra pas être à la vue des contraintes C1 et C2.

Exemple 2. *Supposons qu'il soit possible de munir l'utilisateur final d'un élément sécurisé, e.g. une carte à puce. Alors, il suffirait d'y stocker une clé cryptographique. Á chaque authentification, le terminal $\mathcal{T}$ interroge alors l'élément sécurisé pour permettre le déroulement d'une authentification telle que décrite à la Figure 2.1 .*

La contrainte C1 empêche le déroulement d'un scénario tel que celui décrit ci-dessus tandis que la contrainte C2 empêche un scénario similaire à l'exemple suivant.

Exemple 3. *Afin d'authentifier de manière certaine l'utilisateur, equensWorldline l'enrôle récoltant toutes ses données identifiantes et les stocke sur ses serveurs. Lorsqu'une phase d'authentification s'engage, il récupère toutes les valeurs d'empreintes de l'utilisateur à l'instant t et les compare à celles précédemment stockées.*

D'un point de vue sécuritaire, ce dernier exemple serait acceptable (en considérant une base de données inattaquée) mais le respect de la vie privée de l'utilisateur ne serait alors pas assurée.

2.3 Description du stade TA1

Selon le mode de fonctionnement de TA, un utilisateur souhaitant s'authentifier va prouver la connaissance d'un secret, difficile à retrouver pour toute entité autre que lui 2.1. Pour ce faire, plusieurs problématiques affleurent :

1. **Secret Cryptographique.** Pour être considéré comme un *secret cryptographique*, une donnée doit apparaître aléatoire (ou pseudo-aléatoire) et uniformément distribuée. Un secret cryptographique consistera en une telle donnée et son imprédictibilité découlera de sa longueur (environ 100 bits d'après l'ANSSI en 2016 [112]) pour assurer un niveau de sécurité adéquat. Les empreintes biométriques sont des secrets contenant une certaine quantité d'entropie.

2. **Gestion de secret.** Une fois qu'un tel secret cryptographique a été généré, se pose la question de sa gestion. Si un secret cryptographique est conservé en clair sur un dispositif, il peut être lu par un attaquant. S'il est stocké, il doit l'être dans un élément sécurisé.

3. **Reproduction de secret.** Au vu de notre contexte purement logiciel (contrainte C1), aucun secret ne peut être stocké ; il faudra être en mesure de la reconstruire à partir de certaines données plus à même d'être conservées ou récupérées.

2.3.1 Aperçu du fonctionnement

Initié en 2008 pour répondre à l'appel à projet rappelé ci-dessus, la solution TA a pour philosophie de générer à la demande la clé secrète sk d'un utilisateur $\mathcal{U}$. Ainsi, l'idée consiste à générer cette clé à partir de données identifiantes que l'utilisateur aura a sa disposition ; ces données seront alors portées par l'utilisateur $\mathcal{U}$ et/ou ses appareils $\mathcal{T}$ (Smartphone, PC, ...) vus comme des sources d'aléa nécessaire à la génération du secret sk. Cette démarche correspond au domaine du *fingerprinting* pour lequel des études formelles ont été proposées en 2010 et en 2016 respectivement dans le cas des navigateurs [110] et des smartphones [111]. Dans le cas des empreintes biométriques physiologiques classiques, deux faits sont communément admis :

- chaque utilisateur peut être identifié uniquement par sa valeur d'empreinte (*e.g.* empreinte digitale) ;
- les variations dans l'acquisition de tels signaux empêchent l'utilisation de techniques classiques (fonction de hachage, chiffrement) dans leur exploitation.

Les choses sont un peu différentes dans le cas des empreintes de navigateur et d'appareils : c'est l'accumulation de valeurs de plusieurs champs caractéristiques (*e.g.* liste des plugins, user agent, système d'exploitation) qui constituera une empreinte de navigateur [113, 114]. Chaque champ apporte de l'information mais pas suffisamment pour identifier uniquement un utilisateur ; on parle d'identification partielle. Certains travaux [115] proposent alors d'appliquer une fonction de hachage sur l'ensemble de ces valeurs pour obtenir un identifiant secret relatif à un profil. Dans toute la suite de ce manuscrit, chaque champ apportant de l'information sur un utilisateur sera défini comme un *attribut* : la liste des plugins, le user agent, le système d'exploitation, numéro de SIM sont autant d'attributs permettant d'identifier partiellement des utilisateurs. Anticipant l'émergence du *fingerprinting*, la solution TA proposait d'appliquer un KDF à de telles données pour générer le secret sk. Avec une limitation cependant : les valeurs considérées ne doivent souffrir d'aucune variation puisqu'une modification d'un seul champ modifierait alors la valeur du secret retourné.

Pour résumer, l'idée générale de TA est donc la suivante : $\mathcal{U}$ muni de $\mathcal{T}$ doit être capable de générer et re-générer à la demande un secret sk nécessaire à son authentification comme illustré par la Figure 2.1. Cette génération de secret pourra tirer profit de ses différents attributs.

Afin d'assurer la sécurité du secret généré, l'agrégation d'un certain nombre d'attributs constituant un *profil utilisateur* pourra être réalisée.

2.3.2 Profil utilisateur

L'IMEI (pour *International Mobile Equipment Identity*) est un attribut tel que défini ci-dessus. Composé de 15 digits dont 14 aléatoirement choisis, il permet d'identifier de manière unique chacun des terminaux de téléphonie mobile et répond donc au facteur de possession FA-2 (Définition 43). Cet attribut donne accès à $14 \times log_2(10) \approx 47$ bits d'entropie ; ce sera l'un des attributs portant le plus d'attributs mais ce sera bien évidemment trop peu d'un point de vue cryptographique.

L'idée est donc de considérer un *profil utilisateur*, noté Ω, qui consistera en l'ensemble des valeurs d'attributs récoltés par l'application logicielle App. Un exemple de profil utilisateur tel que défini au stade TA1 est représenté en Figure 2.2.

Attribut att_i	Valeur ω_i
IMEI	354784054129987/02
SIM	208 20 1134567890
Connecteur USB	micro USB
Définition écran	720×1280
Debogueur	Android Debug Bridge (ADB)
Hash de la librairie OpenSSL	8e52d5cb...20afb443
Mode root	oui / non
Nom du téléphone	Téléphone de Bob
WiFi activé	oui / non
Hash de l'application App	0d04bfeb...34c9984d

Figure 2.2: Profil Utilisateur au stade TA1

Renseigné par une politique, un module dédié de la solution TA aura pour but de collecter le profil Ω en s'assurant que la nature des attributs considérés permettra de réaliser une authentification TA (Définition 44). Le cadre de nos travaux consistera à en extraire un secret stable sk pour permettre de réaliser cette authentification. D'autre part, pour assurer un certain niveau de sécurité, un profil utilisateur devra porter assez d'entropie afin de générer une clé sk de longueur suffisante [112]. Alors que cette estimation est plutôt directe dans le cas d'attributs comme ceux spécifiés par la Figure 2.2, nous verrons au Chapitre 3 que certains sont plus difficiles à évaluer ; nous verrons que cette évaluation fera l'objet d'études futures pour permettre l'accession au stade TA3-Ind.

Nous présentons alors le fonctionnement de notre solution au stade TA1.

2.3.3 Première solution

L'objectif O4 requiert que l'utilisateur soit capable, à partir de son profil Ω, de générer un secret sk lors de chaque session d'authentification. Notons $\Omega = (w_1, \ldots, w_n)$, où w_i est la valeur de l'attribut att_i, salt une graine publique, KDF un KDF et $(\mathsf{authGen}, \mathsf{authProve}, \mathsf{authVerif})$ un protocole d'authentification tel qu'introduit à la Figure 2.1. Les primitives d'enrôlement et d'authentification de la solution TA1 peuvent être respectivement décrits à travers les Figures 2.3 et 2.4.

Entrée : Un profil $\Omega = (w_1, w_2, \ldots, w_n)$.

1. Le serveur $\mathcal{S}$ envoie un challenge c à l'utilisateur $\mathcal{U}$.
2. L'utilisateur $\mathcal{U}$ crée une identité publique à partir de son profil Ω :
   - Il calcule $sk = \mathsf{KDF}(w_1, \|w_2\| \ldots \|w_n\|\mathsf{salt})$.
   - Il génère sa clé publique $\mathsf{pk} \leftarrow \mathsf{authGen}(\mathsf{sk})$
   - Il génère un preuve $\pi = \mathsf{authProve}(\mathsf{sk}, \mathsf{pk}; c)$.
3. $\mathcal{U}$ envoie (pk, π) au serveur $\mathcal{S}$.
4. $\mathcal{S}$ vérifie que $\mathsf{authVerif}(\pi, \mathsf{pk}) = 1$ et associe pk et salt à $\mathcal{U}$. Il abandonne sinon.

Figure 2.3: Enrôlement de $\mathcal{U}$ via Ω au stade TA1

À la fin de l'enrôlement, le serveur d'authentification a alors associé l'identité publique pk à l'utilisateur $\mathcal{U}$. Lors d'une authentification future, ce dernier devra alors prouver la connaissance du secret associé sk pour convaincre de son identité. Ne disposant d'aucune capacité de stockage sécurisé, il n'est pas dans son intérêt de conserver ce secret sur son terminal $\mathcal{T}$. Ainsi, lorsque $\mathcal{U}$ souhaite s'authentifier, son application App va collecter son profil $\Omega' = (w'_1, w'_2, \ldots, w'_n)$ où w'_i consiste en la valeur de att_i au moment de l'authentification. $\mathcal{U}$ suit alors la procédure décrite dans la Figure 2.4 pour tenter reconstruire le secret sk.

Contrairement à la session d'enrôlement, l'utilisateur $\mathcal{U}$ n'envoie pas pk' au serveur $\mathcal{S}$. En fait si Ω' est correct (*i.e.* égal à Ω), la consistance est évidente. Cela entraine que $\mathsf{sk}' = \mathsf{sk}$ et par conséquent $\mathsf{pk} = \mathsf{pk}'$. Finalement, on obtient :

$$\mathsf{authVerif}(\pi', \mathsf{pk}) = \mathsf{authVerif}(\mathsf{authProve}(\mathsf{sk}', \mathsf{pk}'; c') = \mathsf{authVerif}(\mathsf{authProve}(\mathsf{sk}, \mathsf{pk}; c'), \mathsf{pk}) = 1.$$

Est-ce une authentification TA ? En supposant que les profils d'authentification Ω' et Ω soient égaux en tout point, alors un utilisateur est capable de reconstruire le même sk et donc de s'authentifier. Au vu du profil utilisateur (Figure 2.2), les objectifs O2 et O3 (Définition 44) sont clairement réalisés alors que l'objectif O1 n'est pas atteint a priori. Pour ce faire, il est nécessaire au stade TA1 de recourir à une interaction avec l'utilisateur

Entrée : Un profil $\Omega' = (w'_1, w'_2, \ldots, w'_n)$.

1. $\mathcal{S}$ envoie un challenge c' et salt à $\mathcal{U}$.
2. $\mathcal{U}$ doit re-générer sk pour prouver sa connaissance du secret associé à pk
   - Il calcule $\mathsf{sk}' = \mathsf{KDF}(w'_1, \|w'_2\| \ldots \|w'_n\|\mathsf{salt})$.
   - Il génère $\mathsf{pk}' \leftarrow \mathsf{authGen}(\mathsf{sk}')$.
   - Il génère $\pi' = \mathsf{authProve}(\mathsf{sk}, \mathsf{pk}; c')$.
3. Il envoie π au serveur $\mathcal{S}$.
4. $\mathcal{S}a$ vérifie que $\mathsf{authVerif}(\pi', \mathsf{pk}) = 1$. Il rejette l'authentification sinon.

Figure 2.4: Authentification de $\mathcal{U}$ via Ω aux stade TA1

qui doit alors taper son mot de passe, ce qui constitue une une première limitation au stade TA1. Cette limitation est renforcée par la tendance actuelle à se passer de solutions d'authentification basées sur l'usage de mots de passe à l'image de ce que promeut le consortium FIDO constitué de géants du domaine (Google, Qualcomm, Discover Financial Services, Visa Inc, Mastercard, ...) [116, 117]. On notera L0 la limitation qui contraint l'utilisateur à une interaction du type mot de passe.

2.4 Vue d'ensemble de la solution TA et différents composants

Afin de mieux situer les travaux décrits ici au sein du projet TA, nous donnons un aperçu de l'architecture de la solution logicielle App déployée sur les plates-formes Android et iOS principalement.

2.4.1 Application App

Présentée par la Figure 2.5, l'application App met en jeu différents SDKs (*Software Development Kits*) :

- **TA.** Constitue toutes les actions à prendre dans la vie de l'application : interface entre l'application et le monde extérieur, mise en place de canaux de communication ou encore génération et/ou vérification des politiques d'authentification, récupération du profil utilisateur (Figure 2.2) ;
- **HCE.** Pour *Host Card Emulation*, ce SDK permet la gestion de la technologie HCE (sous l'OS Android), permettant à un smartphone d'émuler et de s'affranchir des cartes bancaires pour les paiements de proximité ;

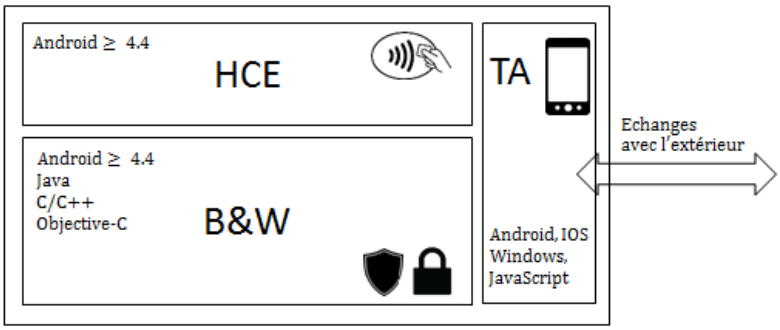

Figure 2.5: Architecture de la solution App

- **B&W.** Le SDK Black&White (B&W) constitue la brique de sécurité de l'application App. Il s'y déroule les opérations cryptographiques dont la génération de clés permettant de se protéger de l'adversaire en boîte noire qui aurait accès aux entrées sorties des différents algorithmes. D'autre part, un système de défense logicielle y est également déployé détectant toute intrusion logicielle.

2.4.2 Zoom sur le SDK B&W et place du module FAST

Garant de la sécurité cryptographique et logicielle de l'application App, nous présentons les différents composants du SDK B&W et leur articulation à travers la Figure 2.6.

- **IDE.** Pour *Intrusion Detection Engine*, ce module met en jeu différentes techniques sécuritaires logicielles telles que la mise en place de détecteurs s'assurant d'un fonctionnement légitime (le mode root est-il activé ?, le programme est-il émulé ?, suit-il une succession d'étapes attendue ? ...) et de protections associées telles que de l'obfuscation logicielle;

- **CryptoCore.** Ce module contient tous les algorithmes cryptographiques impliqués dans la solution TA (signature et MAC, chiffrement, hachage, génération de clés, ...). Ces algorithmes sont résistants à un attaquants en boîte noire. En particulier, c'est en son sein qu'a lieu la génération du secret sk à partir du profil Ω tel que défini par les Figures 2.3 et 2.4.

- **Whiteboxes.** Selon le modèle de l'attaquant en boîte blanche (détaillé en Section 2.5 le simple chargement en mémoire d'une donnée sensible (*i.e.* une clé cryptographique) la rend visible par un attaquant. Bien qu'un pan de la littérature se penche sur ces problématiques, on reste loin d'applications industrialisables.

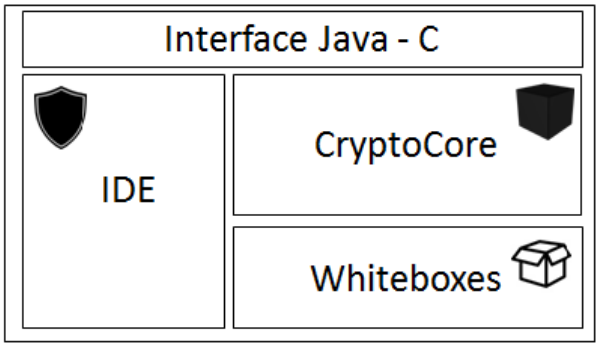

Figure 2.6: Architecture du SDK B&W

Ainsi, des solutions *ad hoc* sont proposées ; à l'heure actuelle, equensWorldline achète de telles solutions à un fournisseur privé Soc [1].

Place du module FAST Au stade TA2, un module indépendant sera dédié à la génération de clé ; au vu de la Figure 2.6, ce sera en fait un sous-module du CryptoCore.

2.5 Introduction aux modèles de sécurité

En pratique, les différents audits passés par equensWorldline (PCI, Visa, Mastercard,...) s'attachent à vérifier les bons usages de la cryptographie classique (modèle en boîte noire) et sécuritaire (modèle en boîte blanche). Jusqu'au début des années 90, les schémas cryptographiques proposés respectaient le traditionnel modèle en boîte noire pour lequel les plateformes exécutant les calculs étaient supposées sécurisées. Un adversaire n'avait alors accès qu'aux entrées sorties d'un système mais pas à son état interne [118]. Les travaux de P. Kocher [119] ont exhibé les limitations de telles considérations puisqu'un adversaire aura potentiellement accès à des informations que laissent fuiter des dispositifs physiques. Étendant ces considérations, la cryptographie en boîte blanche a été introduite en 2003 par Chow *et al.* [120]. Le domaine de la cryptographie en boîte blanche peut se voir comme un cas particulier du domaine de l'obfuscation de programme [121] dont les considérations restent théoriques et éloignées de considérations pratiques [122, 123]. Cependant, contrairement à l'obfuscation de programme, la cryptographie en boîte blanche n'a pas pour objectif – même si cela peut en être une conséquence– de rendre inintelligible un programme mais seulement s'assurer que les clés secrètes sont inaccessibles à un attaquant logiciel [124]. D'autre part, les tentatives d'implémentations dans le modèle

[1] Par mesure de confidentialité, nous ne révélons pas le nom dudit fournisseur.

en boîte blanches ont été cassées [125, 124] de sorte que d'un point de vue industriel, les solutions sont généralement protégées par des implémentations *ad hoc*.

Ainsi, equensWorldline considère deux modèles de sécurité dans sa conception de logiciels, :

- **Modèle Blackbox.** Modèle de sécurité classiquement dépeint d'un point de vue cryptographique : l'attaquant $\mathcal{A}$ peut voir les échanges entre l'utilisateur $\mathcal{U}$ utilisant App installée sur $\mathcal{T}$ et un serveur d'authentification. $\mathcal{A}$ aura accès à certains oracles lui permettant de simuler cette vue ;
- **Modèle Whitebox.** Par le requis R1, notre solution doit être logiciellement protégée. Dans ce modèle, un attaquant whitebox $\mathcal{A}'$ peut se voir comme un attaquant logiciel qui a potentiellement accès à "tout ce qui se passe" sur les terminaux $\mathcal{T}$ de l'utilisateur $\mathcal{U}$. Plus précisément, toute donnée chargée en mémoire (*e.g.* clé ou profil utilisateur) par $\mathcal{T}$ est accessible à $\mathcal{A}'$.

2.5.1 Modèle en boîte blanche

Précisions sur la cryptographie en boîte blanche

Comme la cryptographie en boîte noire, la cryptographie en boîte blanche est un domaine à part entière de la cryptographie. Cependant, les techniques proposées dans la littérature sont soit cassées [125, 124] ou pas (encore) applicables au monde industriel [121, 122, 123]. Ainsi, nous procédons aux précisions suivantes :

- le modèle en boîte blanche est commun à la cryptographie académique et aux problématiques industrielles et jouit de modèles proches [118] ;
- les solutions de la littérature ne sont pas utilisables dans le monde industriel ;
- des implémentations d'algorithmes cryptographiques adaptées aux contraintes industrielles et répondant aux contraintes du modèle en boîte blanche existent ; elles sont commercialisées sous le nom de *whiteboxes*.

Ces *whiteboxes* constituent des implémentations *ad hoc* pour lesquelles les principes de Kerckhoffs [21] ne s'appliquent pas. En effet, lors des audits notamment, la consistance des algorithmes est vérifiée sans qu'aucune spécification sur leur fonctionnement interne ne soit requise.

Description du modèle en boîte blanche

Un attaquant logiciel $\mathcal{A}'$ a potentiellement accès à toutes les données chargées en mémoire et pourra même en modifier certaines. Par exemple, il pourra accéder à la valeur d'un attribut. L'accession/modification d'une valeur d'un programme s'appelle un *exploit*. Comme son nom l'indique, un exploit n'est pas simple à réaliser et l'accession par $\mathcal{A}'$ à un attribut n'implique pas que tous les autres attributs seront menacés. Par exemple, en contournant les défenses du module IDE, l'attaquant $\mathcal{A}'$ pourra accéder à une libraire sans pour autant être capable d'apprendre d'autres attributs. Ainsi la réalisation d'un exploit, bien que préoccupante, n'impliquera pas nécessairement la fin de vie d'un profil utilisateur. Dans les faits, il est peu probable qu'un attaquant ait accès à de nombreux attributs sans que l'application App ne s'en aperçoive et ne lance des contre-mesures adaptées via le module IDE notamment.

Aujourd'hui, le fossé entre théorie et pratique est encore trop important -d'un point de vue des temps de calcul- pour appliquer des solutions académiques au monde industriel. Ce sont donc sont des solutions *ad hoc* qui sont utilisées. equensWorldline achète de telles solutions à une société privée Soc, qui permettent de "whiteboxer" certains algorithmes. Plus précisément, étant donné un algorithme cryptographique standard sécurisé du point de vue de $\mathcal{A}$, la solution fournie par Soc permet de le transformer en un algorithme sécurisé du point de vue de $\mathcal{A}'$. Ainsi, quiconque pourra constater la consistance de l'algorithme mais comprendre ou déduire des informations sur son fonctionnement logiciel revient à en casser la sécurité.

2.5.2 Modèle en boîte noire

Comme présenté à travers la Figure 2.5, la mise en œvre de différents SDKs permet l'articulation de l'application App. Le cœur des travaux présentés au prochain chapitre concerne la génération de clés cryptographiques à partir d'attributs utilisateurs. Cette tâche sera effectuée par le module FAST, intégré au CryptoCore. Avant de la formaliser au cours du prochain chapitre, nous considérons que la sécurité de FAST devra résister à un attaquant $\mathcal{A}$ de type blackbox ayant accès à :

1. à ce que l'application App prend en entrée : des données publiques relatives à l'utilisateur $\mathcal{U}$. En revanche, le profil Ω, source de la génération de la clé sk n'a pas vocation à être divulgué ;

2. à ce que l'application retourne : des paramètres publics liés à l'utilisateur et une preuve de connaissance de la clé sk. Parmi ces paramètres publics, certains dénotés hd seront liés à la génération de la clé sk.

Nous proposons alors un zoom sur le module FAST qui :

1. prend en entrée un profil utilisateur Ω (voir Figure 2.2) et des données publiques ; ce profil (secret) est renseigné par le SDK TA tandis qu'il est protégé par le module IDE et des whiteboxes ;

2. génère un secret cryptographique sk, lui aussi protégé par les mêmes outils et une donnée publique hd transmise au serveur d'authentification ;

3. re-génère ledit secret cryptographique sk en s'appuyant sur hd;

Que ce soit par le jeu des mises à jour ou par l'utilisation d'un framework autre que la solution TA, le module FAST sera amené à générer plusieurs fois des clés cryptographiques pour un même utilisateur. De ce fait, de nombreuses données publiques hds seront générées et certaines clés pourraient être attaquées par des exploits de $\mathcal{A}'$ ou plus simplement par des failles de sécurité d'applications autres que TA utilisant FAST.

Ainsi, le modèle de sécurité associé, formalisé au prochain chapitre, requiert que ni les publications des données auxiliaires hds ni d'éventuelles corruptions de clés secrètes sk issues d'un même profil n'impacte pas la sécurité dudit profil et des secrets non corrompus. Un résumé de la situation est proposé en Figure 2.7.

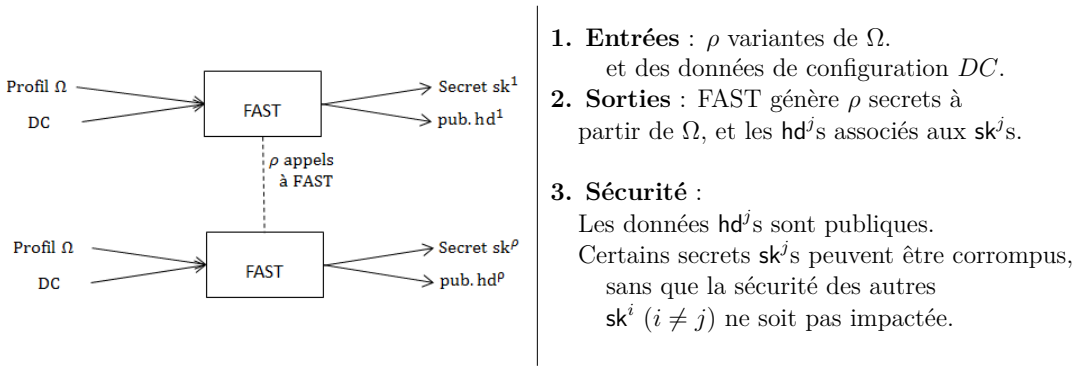

Figure 2.7: Aperçu du modèle de sécurité de FAST

Méthodologie

Pour qu'une solution proposée satisfasse à ces deux modèles de sécurité en même temps, la méthodologie adoptée par equensWorldline est la suivante :

1. Proposer une application App sécurisée vis à vis de l'attaquant $\mathcal{A}$.

2. Utiliser les solutions *ad hoc* fournies par Soc pour le rendre résistant à l'attaquant $\mathcal{A}'$.

Dans la suite e ce manuscrit, nous nous focaliserons donc sur la proposition de solution résistantes à l'attaquant $\mathcal{A}$

2.6 Limitations au stade TA1

En plus de situer poser le cadre de nos travaux, cette description du stade TA1 va permettre d'exhiber certaines limitations dont la résolution constituera le cœr du prochain chapitre.

2.6.1 Limitations L0 et L1 ou le manque d'attributs diversifiés

Les attributs présentés dans le profil utilisateur Ω (Figure 2.2) couvrent les objectifs de type O2 et O3. En effet, les attributs présentés permettent principalement prouver la possession du bon téléphone (IMEI, ...) et des bons logiciels (haché des librairies). Malheureusement, elles ne couvrent pas les facteurs biométriques (FA-3 et FA-4) : ainsi, la seule manière pour que l'objectif global O4 soit rempli -et qu'une authentification TA soit garantie- est de vérifier la connaissance d'un mot de passe par l'utilisateur pour assurer O1 (Limitation L0). Récemment, de plus en plus de solutions d'authentification forte visent à se détacher de la combinaison classique FA-1 + FA-2 telle que décrite dans l'exemple 1. En effet, l'objectif d'un bon nombre de solutions industrielles est de proposer une authentification forte avec une expérience utilisateur améliorée : une solution souvent envisagée est une transaction au cours de laquelle le rôle de l'utilisateur sera moins contraignant que de taper un mot de passe. Ainsi, il est donc requis d'étudier de nouveaux types d'attributs pour pouvoir potentiellement couvrir tous les facteurs d'authentification tout en proposant des solutions plus adaptées [126].

La biométrie comme solution idoine ?

Ainsi les facteurs FA-3 et FA-4 apparaissent particulièrement adaptés aux limitations soulevées. En particulier, le domaine de la biométrie physiologique (Facteur FA-3) voit son intérêt décuplé depuis que de plus en plus de terminaux utilisateurs permettent d'y avoir accès [75, 73, 74].

Cependant, trois limitations sont à prendre compte lorsqu'il s'agit de biométrie. Les deux premières sont d'ordre technologique alors que la dernière relève d'une limitation théorique :

- la solution proposée par equensWorldline doit être utilisable par tous les terminaux du marché. Même si lecteurs d'empreinte digitale, modules de reconnaissance vocale voire lecteurs d'iris se répandent ou tendent à se répandre [75, 76, 77, 74], ces appareils sont les plus chers et on reste loin d'une démocratisation totale. Ainsi d'autres données d'authentification doivent être considérées ;
- d'autre part, l'usage de telles données par lesdits appareils ne correspondent pas à la philosophie de la solution TA. Par exemple, un scan d'empreinte digitale par le

smartphone Samsung Galaxy S6 permet seulement de dire si l'empreinte présentée correspond à l'utilisateur enrôlé. La réponse est donc binaire : oui s'il s'agit de l'utilisateur légitime; non sinon. Il n'est donc pas possible de prouver la validité du signal biométrique considéré. Cela en restreint l'usage à une authentification en local seulement vis-à-vis du téléphone. alors que par concept, notre solution d'authentification vise à générer des clés à partir de données utilisateurs identifiantes pour en prouver la validité auprès d'un serveur d'authentification ;

- Dans l'hypothèse où les terminaux existants permettaient, de manière légitime, d'accéder aux signaux biométriques, demeure la problématique d'authentification à partir de telles données dans le respect de la vie privée. En effet, les empreintes biométriques sont sujettes à des variations intrinsèques. En reprenant l'exemple de l'empreinte digitale, plusieurs scans donneront autant de signaux différents empêchant ainsi l'utilisation de techniques classiques telles que chiffrement et fonction de hachage. Pour permettre une authentification sans ambiguïté, il est nécessaire d'avoir accès à ces signaux en clair.

Ce dernier point soulève deux problématiques fortement liées. En plus de la contrainte sur le respect de la vie privée (contrainte C2), si un serveur d'authentification possédant les signaux biométriques de ses utilisateurs était attaqué, cela remettrait en question la sécurité de tout autre système s'appuyant sur les même signaux biométriques. La littérature regorge de travaux dont le but est d'allier authentification et biométrie dont le domaine des Fuzzy Extractors [85, 13, 83, 94] introduit au Chapitre .

Potentialité et limites des PUFs

Les PUFs, décrits au Chapitre 1 présentent, par manufacture, une source d'aléa potentiellement exploitable. À challenge donné, une telle entité retourne théoriquement la même clé en réponse à une stimulation. Pou répondre aux variations environnementales qui auront tendance à impacter leur stabilité, l'usage de FEs a été proposé. Le secret variable donné en entrée du FE consiste alors la sortie générée par le PUF. Malheureusement, alors que certaines attaques peuvent limiter l'usage des PUFs [127, 128], notre objectif est de nous affranchir le plus possible d'éléménets matériels dans la conception de nos solution. En effet, la solution TA est justement née de ce type de contrainte : si un élément matériel est mis en jeu pour assurer la sécurité de notre solution, il nous faudra alors obtenir l'accord –et donc payer– le constructeur pour l'utiliser. Ce fut le cas avec les opérateurs mobiles dans le cadre des cartes SIM. D'autre part, alors que les constructeurs munissent actuellement leurs smartphones d'environnements de calcul sécurisés ou TEE[129], il n'est pas possible d'y accéder sans leur accord préalable.

Ainsi, nous limitons au maximum l'usage d'éléments matériels (dont equensWorldline ne serait pas le propriétaire) dans la conception de nos solutions.

D'autres sources de données identifiantes

Ainsi, au vu des contraintes C1 et C2 déjà identifiées, il est légitime de préciser que la solution TA doit pouvoir s'instancier sur n'importe quel smartphone. En effet, mêmes si certains appareils gèrent certains signaux biométriques, ce n'est pas encore le cas de la flotte du grand public et des données d'identification alternatives doivent être trouvées. Ainsi, il faudra exhiber des attributs accessibles par l'application App sur n'importe quel terminal et permettant de répondre au facteur FA-3 ou FA-4. Cela conduit à la contrainte suivante :

- **C3.** Un attribut au sens TA doit être accessible de manière logicielle par l'application App installée sur le terminal $\mathcal{T}$ de l'utilisateur $\mathcal{U}$. Cela doit se faire sans déploiement de matériel dédié tel qu'un lecteur d'empreinte digitale ou un module de reconnaissance par exemple.

La biométrie, au potentiel certain n'est pas encore exploitable pour répondre aujourd'hui aux problématiques business d'equensWorldline. Dans toute la suite, le manque d'attributs diversifiés et adaptés sera noté comme étant la limitation L1.

2.6.2 Limitation L2 ou la gestion d'attributs variables

Les empreintes biométriques discutées précédemment sont des données qui peuvent varier entre enrôlement et authentification tout en restant valides. Jusqu'ici, ne son considérés dans le paradigme TA que les attributs présentant des valeurs d'authentification égales au valeurs d'enrôlement (Figure 2.2). Cependant, afin d'augmenter l'entropie contenue dans un profil utilisateur et de diversifier la nature des attributs (Limitation L1), nous allons étudier (Chapitre 3) de nouveaux types d'attributs dont la variabilité rappellera les problématiques liées aux empreintes biométriques. Ainsi, même si la biométrie est laissée de côté car ne satisfaisant pas la contrainte C3, il faudra, pour le stade TA2, proposer une solution d'authentification mettant en jeu de nouveaux types d'attributs (potentiellement variables) tout en satisfaisant la contrainte CP et l'objectif O4. Généralement, il n'est pas évident de générer une clé (objectif O4) à partir de données variables. Nous nous référons à cette problématique par la limitation L2, qui fait écho à celle des Fuzzy Extractors (Chapitre).

2.6.3 Limitation L3 ou l'importance d'un attribut

Par concept, l'authentification TA ainsi proposée (Figures2.3 et 2.4) considère d'importance égale les différents attributs d'un profil utilisateur (Figure 2.2). En effet, hacher la concaténation des valeurs d'attributs implique que ces attributs sont supposés de criticité équivalente : si l'une des valeurs d'attributs diffère, le secret sk ne sera pas retrouvé et l'authentification échouera. Néanmoins, certains attributs apparaissent plus importants que d'autres. Par exemple, l'IMEI doit évidemment être le même entre enrôlement et authentification alors que l'activation du WiFi ne revêt pas, a priori, une importance aussi capitale. L'idée d'introduire une hiérarchie sur les attributs a alors émergé pour proposer deux types de criticité :

- **Fatal.** Si un attribut de type *fatal* ne présente pas une valeur d'authentification jugée comme valide (*i.e.* relativement proche de la valeur d'enrôlement), l'authentification échouera.
- **Warning.** Si un attribut de type warning ne présente pas une valeur d'authentification jugée comme valide, on dira qu'un warning est *levé*. Au-delà d'un certain seuil T de warning levés, l'authentification d'un utilisateur doit échouer.

En fait l'idée est de dire que les attributs de type *fatal* caractérisent pleinement l'utilisateur et que leur non respect doit alors entraîner un échec de l'authentification. D'autre part, les attributs de type *warning* constituent un faisceau de présomption. Si trop d'incohérences y apparaissent, *i.e.* un nombre de warning levés supérieur à T, alors l'authentification échouera.

La solution, telle que proposée au stade TA1 ne permet donc pas de faire le distingo Warning/Fatal. Une première parade serait d'appliquer indépendamment une fonction de hachage sur chaque valeur d'attribut. Ainsi, considérant n attributs, il serait nécessaire de calculer n empreintes d'attributs par l'usage d'une fonction de hachage et de générer les preuves d'authentification associées. Pour gérer les warnings, le serveur d'authentification tolérerait alors des preuves invalides pour au plus T attributs estampillés "warning". Au delà de son inélégance, cette solution présente deux limitations:

- Le serveur apprend quels sont les attributs de type *warning* qui n'ont pas été authentifiés. Cette situation va à l'encontre de a contrainte C2 puisque le serveur apprendrait de l'information sur l'utilisateur ;
- D'autre part, le peu d'entropie contenu dans certains attributs (*e.g.* activation du WiFi, définition de l'écran) ne permet pas une gestion individuelle. En effet, tester de manière exhaustive toutes les valeurs possibles pour une empreinte pourrait

permettre d'en apprendre sa valeur véritable, allant une fois encore à l'encontre de la contrainte C2.

Dans toute la suite, nous nous référerons à la limitation ci-dessus par l'appellation L3.

2.7 Récapitulatif

Ainsi, la conception du stade TA1 a permis d'établir les objectifs O1-O4 découlant de l'objectif OP (authentification forte) déduits de l'appel à projet de la SG. D'autre part, des contraintes respectivement des clients et d'organismes étatiques ont été présentées (C1-C3) pendant qu'émergeaient trois problématiques (P1-P3) et trois limitations (L1-L3) liées. Cela a constitué le cadre dans lequel le stade TA2 a été lancé. À des fins de simplification, nous ne ferons plus de distinctions entre objectif, contrainte, problématique : chaque point sera vu comme un requis. La figure 2.8 constitue un récapitulatif des requis que doit satisfaire la solution TA.

Requis	Définition
R1 (CP)	Solution exclusivement logicielle. Ni stockage sécurisé ni matériel spécifique (*e.g.* ni SIM ni lecteur d'empreintes, voix, ...).
R2 (O1)	Authentifier l'utilisateur $\mathcal{U}$.
R3 (O2)	Authentifier ses terminaux $\mathcal{T}$.
R4 (O3)	Authentifier son application App.
R5 (O4)	Générer un secret R assurant la réalisation de R2, R3, R4.
R6 (C2)	Assurer le respect de la vie privée.
R7 (P1)	Exploiter des sources d'aléa (P1) (génération de R à partir de Ω).
R8 (P2,P3)	Gestion du secret R (re-généré à la demande).
R9	Définition formelle d'un modèle de sécurité.

Figure 2.8: Récapitulatifs des requis de la solution TA

Les requis présentés ici sont tous atteints au stade TA1 mais nous avons vu que la méthodologie mise en œvre soulève des limitations que le prochain chapitre s'attellera à résoudre. Afin de simplifier les références futures à ces limitations, nous les rappelons en Figure 2.9.

Limitation	Définition
L0	Remplir l'objectif O1 sans interaction explicite avec l'utilisateur.
L1	Identifier des attributs plus diversifiés (renforcer R7).
L2	Gestion des attributs variables.
L3	Gestion du type Fatal/Warning.

Figure 2.9: Limitations du stade TA1

Chapitre 3 :

Stade TA2 - Module FAST

Contents

En 2014, il a été décidé la création du module FAST dédié à la génération de clés. Tout en continuant d'assurer les requis de la solution TA (Figure 2.8), ce module devra permettre l'accession au stade TA2 en apportant une réponse aux limitations rappelées à la fin du précédent Chapitre (Figure 2.9). Dans un premier temps, nous nous intéresserons au domaine du *fingerprinting* initié par le stade TA1 -dans le cas d'attributs statiques- et l'étendrons à la résolution des limitations L0 et L1. Nous identifierons alors de nouvelles données d'authentification (L1), permettant pour certaines de s'affranchir de l'interaction

utilisateur (L0). Ensuite nous verrons comment exploiter ces données nouvellement identifiées et potentiellement variables (*e.g.* liste des contacts) afin de répondre à la limitation L2 en nous appuyant sur le domaine des Fuzzy Extractors [13] introduit au chapitre 1. Cette approche, formalisée par FAST, permettra également de répondre à la limitation L3.

Les travaux décrits dans la suite ont fait l'objet d'un dépôt de brevet [3] et d'une participation à l'*Atelier pour la Protection de la Vie Privée* 2014 (APVP 2014) [4].

3.1 Fingerprinting ou réponse aux Limitations L0 et L1

Le but de cette section n'est pas de lister de manière exhaustive les techniques de *fingerprinting* existantes mais de souligner l'importance prise par ce domaine au cours des dernières années. La multiplication des services en ligne et des appareils connectés et personnalisables (*e.g.* Smartphone) permettent d'associer des utilisateurs avec leurs profils numériques.

Une empreinte digitale d'appareil (*device fingerprint* en anglais) est une information collectée sur un dispositif informatique distant à des fins d'identification, d'authentification ou de détection de fraude. Lorsqu'une telle empreinte est récupérée via un navigateur web, on parle d'empreinte navigateur qui est donc récoltée via du *fingerprinting* de navigateur (*browser fingerprinting*) [110]. Ces empreintes digitales peuvent être utilisées pour identifier totalement ou *partiellement* un internaute, un utilisateur ou un appareil même lorsque les témoins (cookies) sont désactivés. Identifier *partiellement* revient à classer un utilisateur parmi un groupe aux caractéristiques communes.

3.1.1 Bref état de l'art

Bien que les techniques de *fingerprinting* ont sûrement été mises en œuvre depuis de nombreuses années par les industriels dans le but, essentiellement, de détecter des fraudes ou de proposer des publicités ciblées aux utilisateurs, ce n'est qu'en 2010 qu'une première étude publique a été fournie par Eckerlsey [110]. Ces travaux ont posé les bases du *fingerprinting* de navigateur. Le site web mis à disposition des utilisateurs [113] propose de calculer l'entropie contenue dans leur navigateur client à travers la collecte de différentes valeurs comme la liste des plugins navigateur, la version navigateur web ou encore l'ensemble des polices présentes. C'est l'agrégation de toutes ces informations qui permet de définir l'empreinte de navigateur contenant une certaine entropie. L'entropie des empreintes navigateur telles que collectées par [110] était alors d'environ 18 bits d'où

la notion d'identification partielle. Par la suite, de nombreux travaux [130, 131, 132, 133] ont permis d'augmenter l'entropie des empreintes de navigateur via l'utilisation du *canvas fingerprinting* notamment [131, 133], qui à elle seule permet un gain de 5 bits d'entropie. D'autre part, d'autres technologies permettent de renforcer le domaine du *fingerprinting* comme le concept d'"evercookies" qui consiste en des cookies capable de se re-générer après avoir été détruits par l'utilisateur [134, 135]. En réponse à ces techniques potentiellement intrusives, le domaine "Web Privacy Measurement" a été introduit et investigué pour répondre aux violations de la vie privée [136, 137]. Plus précisément, l'idée est de détecter et répertorier les techniques de traque et de *fingerprinting* pour compenser l'asymétrie entre le peu de connaissances du grand public et les pratiques utilisées par les sites web. Cependant, ces techniques de *fingerprinting* sont destinées à être utilisées par des ordinateurs de bureau. En effet, les navigateurs web des smartphones ne sont guère personnalisables et il est plus difficile d'en extraire des empreintes ("harder to fingerprint than desktop browsers" [110]). De plus, les utilisateurs de smartphone privilégient bien souvent l'usage d'applications dédiées à la réalisation de leur tâches réduisant d'autant plus l'impact du browser *fingerprinting* dans ce cas précis.

Par la suite, de nombreuses études se sont donc concentrées sur le *fingerprinting* de mobiles. Ces techniques se basent notamment sur les caractéristiques de certains composants comme la caméra [138, 139] ou le micro [140, 141, 142] pour en conclure le modèle du smartphone concerné. Certaines études basées sur l'accéléromètre ont également été proposées [140, 143]. À travers l'étude de réponses caractéristiques, ces résultats permettent identifier l'appareil mis en jeu ; cependant il n'est pas possible d'identifier (au moins partiellement) l'utilisateur comme c'est le cas pour le *fingerprinting* de navigateur. Dans le but de reconnaître des utilisateurs plutôt que leurs appareils, des études s'appuyant sur la liste des applications installées ont été récemment proposées [144, 145]. En 2016, Kurtz *et al.* ont proposé une étude similaire à [110] orientée sur les smartphones tournant sous le sytème d'exploitation iOS. Les attributs considérés prennent en compte bon nombre d'indices de personnalisation parmi lesquels que le choix de la langue, les chansons les plus écoutées, les permissions accordées par l'utilisateur et permettent ainsi une identification unique. Pour permettre une identification unique, il est important de savoir faire la distinction entre les nouveaux utilisateurs et des utilisateurs déjà connus du systèmes pour lesquels certaines valeurs d'empreintes auraient légèrement changé au cours du temps. C'est la problématique de ré-identification [110, 111]. En basant leurs observations sur une base de données de 13 000 empreintes, les travaux présentés par Kurtz *et al.* sont capables de ré-identifier les utilisateurs déjà rencontrés avec une précision de 97%. Comme remarqué dans [111], le système d'exploitation iOS bloque beaucoup plus d'accès que le système Android ; il est donc légitime de considérer que de tels résultats sont encore plus forts sur des smartphones tournant sous Android

beaucou plus permissif.

3.1.2 Mise en perspective

Lien avec la biométrie classique

Les systèmes biométriques, en particulier les empreintes digitales, font référence à des systèmes qui permettent de reconnaître des individus en se basant sur des caractéristiques physiologiques et/ou comportementales. Pour être considérées comme des traits biométriques, ces données doivent respecter les principes suivants [146] :

- **Unicité.** Pour une empreinte donnée (*e.g.* empreinte digitale), chaque individu doit présenter une valeur d'empreinte unique. Cette caractéristique peut aussi se définir par le terme de *diversité* : une empreinte donnée doit assurer une champ de valeurs suffisamment diversifié ;
- **Collectabilité.** Une empreinte biométrique doit pouvoir être récoltée ;
- **Permanence.** La valeur d'empreinte d'un utilisateur doit rester stable au cours du temps.

Une des problématiques de l'exploitation de la biométrie est la reconnaissance d'une empreinte dans le respect de la vie privée. En effet, pour utilisateur donné, deux signaux d'une même empreinte seront assez proches pour l'identifier uniquement mais présenteront toujours d'inhérentes variations. De ce fait, l'usage de techniques cryptographiques classiques (chiffrement, hachage , preuve de connaissance, ...) ne pourra s'y appliquer.

Parallèle avec la solution TA

La notion de collectabilité se retrouve dans nos travaux à travers le requis R1 (Figure 2.8) puisque les données que nous considérons doivent être accessibles par l'application App sans déploiement de matériel dédié. Aussi, ces trois notions peuvent être étendues aux empreintes d'appareils. En reprenant la terminologie propre au projet TA, le terme empreinte ou "fingerprint" présent dans la littérature est désormais remplacé par le terme "attribut":

- **(Non-)unicité.** Deux utilisateurs ou deux devices peuvent avoir des valeurs d'attributs égales. Par exemple, une version de système d'exploitation sera commune à un grand nombre d'utilisateurs. Pour distinguer de manière univoque les utilisateurs, on considérera alors l'entropie totale contenue dans un profil ;
- **Collectabilité.** Un attribut est *collectable* s'il est accessible par l'application App ;

- **Permanence.** Certaines valeurs d'attributs sont amenées à varier et ont donc des durées de vie plus courtes que des empreintes biométriques classiques. Par exemple, il est raisonnable d'estimer que la version du système d'exploitation à une durée de vie de plusieurs mois au moins. A contrario, une liste de cookies peut varier rapidement en fonction de la navigation de l'utilisateur.

De plus en plus d'appels à projets reçus par equensWorldline stipulent de procéder à du *fingerprinting* pour authentifier un utilisateur tout comme certains papiers blancs (*whitepapers*) d'acteurs industriels majeurs [126]. Anticipant cette émergence, les travaux TA ont été lancés en 2008 dans le cadre d'attributs statiques. Au cours de cette thèse, cette démarche s'est poursuivie par l'exploitation d'empreintes d'appareils variables telles que les listes d'applications installées, de contacts, ou de chansons en s'appuyant sur le fait qu'un utilisateur va nécessairement personnaliser ses appareils. Cette entreprise est confirmée par les récents travaux de l'état de l'art qui soulignent les potentialités du *fingerprinting* qui permet dans certains cas d'identifier uniquement un utilisateur. Cependant lesdites études [144, 145, 111] exhibent les potentielles menaces au regard du respect de la vie privée sans en tirer les potentiels bénéfices que pourrait en tirer l'utilisateur. Dans le contexte TA, l'utilisateur et ses appareils $\mathcal{T}$ seront protégés du monde extérieur par les modules IDE et *whiteboxes* (Figure 2.6) tandis que FAST va permettra des clés à partir de empreintes d'appareil identifiantes. Ce faisant, nous apportons un début de réponse aux limitations L0 et L1.

Dans la prochaine section, nous proposons un travail de formalisation préalable à la définition du module FAST.

3.2 Formalisation et concepts

Dans cette section, nous définirons les notions d'*attributs*, de *profil utilisateur* et de *politique d'authentification* pour finalement définir le module FAST via le triplet d'algorithmes (GenParam, GenKey, RepKey). Un attribut correspondra à toute donnée apportant de l'information sur un utilisateur $\mathcal{U}$ et/ou ses terminaux $\mathcal{T}$ de sorte qu'un vecteur d'attributs constituera un profil utilisateur Ω. Selon le contexte et le niveau de sécurité requis, le choix des attributs constituant un profil permettra d'atteindre un niveau d'entropie suffisant.

3.2.1 Attributs et Profil Utilisateur

Jusqu'ici, nous avons confondu attributs et identifiants d'attributs. En pratique, la chaîne de caractères "IMEI" constitue l'identifiant de l'attribut alors que "354784054129987/02"

en sera sa valeur. Les identifiants constituent un ensemble voué à croître à mesure que le domaine du *fingerprinting* et que différents appareils connectés arrivent sur le marché.

Notation Notons désormais $\mathcal{I} \subset \mathcal{P}(\{0,1\}^*)$ l'ensemble des identifiants d'attributs.

Pour pouvoir juger de la similarité entre deux valeurs d'attributs w et w' pour un identifiant donné $\mathsf{id} \in I$, nous allons devoir les comparer. Ains, ces valeurs d'attributs doivent appartenir à un même à un espace métrique.

Notation Pour tout id $\in \mathcal{I}$, il existe un espace métrique associé $\mathcal{M}_{\text{id}}$ contenant toutes les valeurs potentiellement prises par id.

Exemple 1. *"Liste de contacts" et "IMEI" sont des éléments de $\mathcal{I}$ dont les espaces métriques associés sont respectivement $(\mathsf{SDif}_s(\{0,1\}^*), d_S)$ et $(\{0,1\}^{15}, d_H)$.*

Nous proposons alors la définition suivante.

Définition 45 (Attribut et Profil Utilisateur)**.** *Soit $n \in \mathbb{N}$. En utilisant la notation définie ci-dessus, nous définissons :*

1. ***(Attribut)** Un attribut est un triplet $\mathsf{att} = (\mathsf{id}, \omega, b) \in \mathcal{I} \times \mathcal{M}_{\mathsf{id}} \times \{0,1\}$ où :*
   - *$\mathsf{id} \in \mathcal{I}$ est l'identifiant de att ;*
   - *$\omega \in \mathcal{M}_{\mathsf{id}}$ est la valeur prise par att ;*
   - *$b \in \{0,1\}$ est la variabilité de l'attribut :*
     - *$b = 0$ si l'attribut est statique. Aucune variation n'est attendue au cours du temps (e.g. IMEI) ;*
     - *$b = 1$ sinon. L'attribut est potentiellement variable (e.g. Liste de contacts).*
2. ***(Profil Utilisateur)** On définit profil utilisateur comme étant un vecteur d'attributs. Plus précisément, le profil utilisateur dénoté Ω se définit comme le vecteur $\Omega = (\mathsf{att}_1, \mathsf{att}_2, \ldots, \mathsf{att}_n)$.*

Autrement dit, un attribut statique est un attribut pour lequel la valeur d'authentification w' devra être exactement égale à la valeur d'enrôlement w afin d'être reconnu comme valable. *A contrario*, un attribut variable pourra présenter de mineures variations et être reconnu selon la tolérance accordée par la politique d'authentification définie dans la prochaine sous-section.

Notation On notera $\mathcal{ATT}$ l'ensemble des attributs, $\mathcal{ATT}^*$ l'ensemble des profils utilisateurs et $\mathcal{ATT}^n$ l'ensemble de tels profils de longueur n. Dans tout ce Chapitre, n désignera donc la longueur d'un profil utilisateur.

Par abus de langage et lorsque cela ne nuira pas à la compréhension, nous ferons la confusion entre attribut et identifiant d'attribut.

À partir d'un profil Ω et de ses potentielles variations Ω', le module FAST devra générer et re-générer une clé stable en respectant une politique d'authentification, notion que nous définissons ci-dessous.

3.2.2 Politique d'authentification à deux niveaux

En accord avec ses clients (énoncés en 2.2.1), les serveurs Worldline décident des règles que devra suivre un utilisateur $\mathcal{U}$ muni de $\mathcal{T}$ afin de pouvoir être authentifié : c'est la politique d'authentification.

L'idée est que cette politique soit décidée par le serveur tout en étant exécutée localement sur $\mathcal{T}$ via l'application App. Plus concrètement, la politique d'authentification stipule quels attributs sont à sélectionner pour constituer un profil utilisateur par rapport au niveau de sécurité λ, le nombre d'erreurs qui seront tolérées pour chaque attribut variable et indique les caractères *Warning*/*Fatal* des attributs considérés. Considérant un profil $\Omega = (\mathsf{att}_1, \ldots, \mathsf{att}_n)$, la politique d'authentification agit sur deux niveaux :

1. **Niveau 1.** Le niveau 1 concerne les valeurs de chacun des attributs du profil. Plus précisément, il précise le nombre maximum d'erreurs tolérées entre les valeurs d'authentification w'_i et d'enrôlement w_i pour considérer pour considérer valide w'_i ;

2. **Niveau 2.** Le niveau 2 spécifie comment sont estampillés les attributs (Warning ou Fatal) :

   - *Fatal.* la valeur d'authentification d'un attribut Fatal doit respecter la tolérance du niveau 1. Sinon, un échec d'authentification est retourné ;
   - *Warning.* On dit qu'un *warning* est levé lorsque la valeur d'authentification d'un attribut Warning ne respecte pas la tolérance du niveau 1. Lorsqu'un trop grand nombre de *warning* sont levés, l'authentification échoue.

Proposer un module respectant les niveaux 1 et 2 de la politique d'authentification revient à répondre aux limitations L2 et L3. La description du stade TA1 proposée au chapitre précédent (Figures 2.3 et 2.4) requiert une exacte égalité entre les profils d'enrôlement Ω et d'authentification Ω' pour permettre une authentification de l'utilisateur $\mathcal{U}$ muni de $\mathcal{T}$. Pour permettre plus de finesse dans la gestion des profils utilisateurs, nous proposons la définition suivante.

Définition 46 (Politique d'authentification)**.** *Soit* $n \in \mathbb{N}$. *On dit que* $(\mathsf{Id}, t, b^2) \in \mathcal{I}^n \times \mathbb{N}^{n+1} \times \{0,1\}^n$ *est une politique d'authentifcation si les conditions suivantes sont satisfaites:*

1. $\mathsf{Id} = (\mathsf{id}_1, \mathsf{id}_2, \ldots, \mathsf{id}_n)$ *est un vecteur d'identifiants d'attribut tel que pour tout* $i \neq j$, $\mathsf{id}_i \neq \mathsf{id}_j$ *;*

2. $t = (t_1, t_2, \ldots, t_n, t_{n+1})$ *est le vecteur du nombre d'erreurs tolérées par attribut. Pour tout* $1 \leqslant i \leqslant n$, $t_i \in \mathbb{N}$ *est le nombre de différences tolérées entre les valeurs d'authentification et d'enrôlement pour l'attribut spécifié par* id_i *(gestion du niveau 1).*

3. $b^2 = (b_1^2, b_2^2, \ldots, b_n^2)$. *Pour tout* i, b_i^2 *indique si l'attribut considéré est de type Fatal* ($b_i^2 = 0$) *ou Warning* ($b_i^2 = 1$) *(gestion du niveau 2).*

4. $t_{n+1} < N \in \mathbb{N}$ *est le nombre maximal de warnings qui peuvent être levés avant que l'authentification n'échoue.*

Afin de déterminer s'il est cohérent pour un profil ou une distribution de se référer à une à une politique d'authentification PA, nous proposons la définition suivante.

Définition 47. *En reprenant la notation de la définition précédente, notons* $\mathsf{PA} = (\mathsf{Id}, t, b^2)$ *une politique d'authentification où* $\mathsf{Id} = (\mathsf{id}_1, \ldots, \mathsf{id}_n)$. *Soit* $\Omega = (\mathsf{att}_1, \ldots, \mathsf{att}_n)$ *un profil de longueur* n *tel que pour tout* $1 \leqslant i \leqslant n$, $\mathsf{att}_i = (\mathsf{id}_i^*, w_i, b_i)$. *On dira que :*

- Ω *est adapté à* PA *si* $\forall i, \mathsf{id}_i = \mathsf{id}_i^*$ *;*
- *Soit* W *une distribution de probabilités, on dira que* W *est adaptée à* PA *si tout* $\Omega \xleftarrow{\$} W$ *est adapté à* PA *;*
- *une famille de distribution de probabilités* $\mathcal{W}$ *est dite adaptée à* PA *si toute distribution* $W \in \mathcal{W}$ *est adaptée à* PA.

Notation Γ désigne l'ensemble des politiques d'authentification. Notons LW et LF les ensembles des identifiants estampillés respectivement *warning* et *fatal*:

$$\mathsf{LW} \stackrel{\text{def}}{=} \{i \in \mathbb{N}, b_i^2 = 1\} \text{ et } \mathsf{LF} \stackrel{\text{def}}{=} \{i \in \mathbb{N}, b_i^2 = 0\}.$$

Remarque 2. *Dans cette remarque, nous précisons plusieurs points :*

- *un attribut de nature variable (Définition 45) pourra potentiellement se voir imposé une tolérance de 0 erreur par le niveau 1 de la politique d'authentification;*

- *un attribut de type statique ne peut pas se voir conférer, par le niveau 1 de la politique d'authentification, une tolérance d'erreurs autre que 0 ;*
- *un attribut de type Fatal n'est pas nécessairement statique et réciproquement.*
- *un attribut de type Warning n'est pas nécessairement variable et réciproquement.*

Nous définissons désormais le fait qu'un profil d'authentification Ω' pourra être reconnu comme dérivant d'un profil d'enrôlement Ω par rapport à une politique d'authentification PA.

Définition 48. *Soient* $\Omega = (\mathsf{att}_1, \mathsf{att}_2, \ldots, \mathsf{att}_n)$ *et* $\Omega' = (\mathsf{att}'_1, \mathsf{att}'_2, \ldots, \mathsf{att}'_n)$ *deux éléments de* $\mathcal{ATT}^n$ *où pour tout* $1 \leqslant i \leqslant n$, att_i *et* att'_i *s'écrivent* $\mathsf{att}_i = (\mathsf{id}_i, w_i, b_i)$ *et* $\mathsf{att}'_i = (\mathsf{id}_i, w'_i, b_i)$. *Soit* $\mathsf{PA} = (\mathsf{Id}, t, b^2) \in \Gamma$ *une politique d'authentification adaptée à* Ω *et* Ω'. *On dira que* Ω *et* Ω' *sont compatibles selon* PA *si :*

1. $\forall i \in \mathsf{LF}$, $d(w_i, w'_i) \leqslant t_i$;
2. $\forall i \in \mathsf{LW}$, $|\{i \in \mathsf{LW} : d(w_i, w'_i) > t_i\}| \leqslant t_{n+1}$.

Notation Étant donnés Ω et Ω' compatibles selon PA, on notera $\Omega \approx_{\mathsf{PA}} \Omega'$.

3.3 Définition du module FAST et modèle de sécurité

Dans cette section, nous nous attachons à définir le module FAST qui pourra se voir comme un générateur de clés prenant en entrée des profils utilisateurs Ω (Définition 45). Nous rappelons les limitations que ce module devra résoudre :

- gestion d'attributs variables et donc de profils variables (Limitation L2) ;
- respect de la politique d'authentification sur deux niveaux (Limitation L3).

Au passage, le module FAST permettra d'exploiter le *fingerprinting*, identifié comme réponse aux limitations L0 et L1

3.3.1 Définition et discussion

Défini par le triplet de primitives $(\mathsf{GenParam}, \mathsf{GenKey}, \mathsf{RepKey})$, le module FAST va générer une politique d'authentification adaptée au niveau de sécurité λ par le biais de la primitive $\mathsf{GenParam}$. La primitive GenKey prendra en entrées un profil utilisateur *adapté* à la politique PA pour générer une clé sk et une donnée publique d'aide de reconstruction hd. Plus tard, sans qu'aucun stockage confidentiel ne soit nécessaire, la primitive RepKey

prendra en entrée un profil Ω' toujours adaptée à la politique d'authentification PA et la donnée publique hd pour reconstruire le secret sk à condition que Ω et Ω' soient compatibles selon PA. Afin de définir le champ d'action dun module FAST, nous considérerons des distributions de probabilités dont seront issus les profils utilisateurs. Nous proposons la définition suivante.

Définition 49. *Soit $\mathcal{W}$ une famille de distributions de probabilités définies sur $\mathcal{ATT}^n$. Un module FAST est un triplet de procédures randomisées (GenParam, GenKey, RepKey) satisfaisant les propriétés suivantes :*

1. *La procédure de génération des paramètres GenParam prend en entrée un paramètre de sécurité λ pour retourner les paramètres publics du système constitués du triplet $(s, l, \epsilon) \in \mathbb{N}^2 \times \mathbb{R}$, de la politique d'authentification $PA \in \Gamma$ et éventuellement de données de configuration $DC \in \{0,1\}^*$.*

2. *La procédure de génération de clés GenKey en prend en entrées $\Omega \in \mathcal{ATT}^n$ adapté à PA, ladite politique $PA \in \Gamma$ et $DC \in \{0,1\}^*$ pour retourner une chaîne $sk \in \{0,1\}^l$ et une donnée de reproduction $hd \in \{0,1\}^*$.*

3. *La procédure de reproduction RepKey prend en entrées $\Omega' \in \mathcal{ATT}^n$ adapté à la politique d'authentification $PA \in \Gamma$ et les chaînes binaires DC et hd. La propriété de correctness de FAST garantit que si $\Omega' \approx_{PA} \Omega$, $(sk, hd) \leftarrow GenKey(\Omega, PA, DC)$ et $((s, l, \epsilon), PA, DC) \leftarrow GenParam(1^\lambda)$, alors $RepKey(\Omega', PA, DC, hd) = sk$. Dans tout autre cas, aucune garantie n'est assurée sur la sortie sk.*

4. *La propriété de sécurité doit garantir que la clé extraite est effectivement pseudo-aléatoire. Plus précisément, pour tout profil Ω issu d'une distribution $W \in \mathcal{W}$, la valeur sk doit appraître pseudo-aléatoire, même en présence de hd. Autrement dit, pour toute distribution $W \in \mathcal{W}$, en notant SK et HD les distributions induites par GenKey (i.e. $(SK, HD) \leftarrow GenKey(W, PA, DC)$), on aura $\delta^{\mathcal{D}_s}((sk, HD), (U_l, HD)) \leqslant \epsilon$.*

Sur l'entropie et la sécurité d'un profil

En fonction du niveau de sécurité visé λ, la politique d'authentification générée PA (primitive GenParam) sélectionnera des identifiants d'attributs pour constituer des profils contenant suffisamment d'entropie relativement à λ. Par exemple, si deux utilisateurs $\mathcal{U}$ et $\mathcal{U}^*$ venaient à présenter des profils Ω et Ω^* compatibles selon PA, la consistance de FAST permettait alors à $\mathcal{U}^*$ de s'authentifier au nom de $\mathcal{U}$. Il faudra alors que l'entropie présente dans les attributs sélectionnés par PA soit suffisante pour authentifier les utilisateurs uniquement.

Ainsi et de manière analogue à la biométrie classique, les attributs d'un utilisateur devront être protégés en confidentialité et intégrité pour assurer la sécurité du système et maintenir le respect de la vie privée. Ce sont des contre-mesures logicielles présentes dans le module IDE (pour *Intrusion Detection Engine*, Figure 2.6) qui nous assurerons que le module FAST prendra en entrée un profil utilisateur supposé ni corrompu ni connu par un attaquant. Par analogie avec le cas biométrique, la connaissance de ce profil correspondrait au cas où un attaquant connaît l'empreinte digitale d'un utilisateur pour qui il essaie de se faire passer.

Propriétés de sécurité

À la manière des Fuzzy Extractors, FAST prend en entrée des données potentiellement variables pour retourner non seulement un secret cryptographique sk mais également une donnée publique hd permettant la reconstruction future du secret sk. Ainsi, les notions de sécurité que nous allons considérer sont équivalentes à celles des Fuzzy Extractors :

1. **Secret cryptographique.** Au vu de la définition introduite ci-dessus, la donnée sk est un secret cryptographique et doit donc apparaître aléatoire aux yeux de tout entité limitée en puissance de calcul. Ainsi, hd généré également à partir de Ω ne devra pas d'information ni sur le profil utilisateur Ω ni sur le secret sk. Puisque sk est issu de Ω, s'assurer en particulier que hd ne révèle rien sur sk suffit. Cela correspond à la propriété de sécurité de la Définition 49.

2. **Réutilisabilité.** La primitive GenKey pourra être appelée plusieurs fois au cours de la vie d'un profil utilisateur puisque:

   - un utilisateur pourra, en fonction de mises à jour de l'application (Application App) ou autres (*e.g.* Système d'exploitation) procéder à un ré-enrôlement de son profil utilisateur;
   - la brique FAST est un générateur de clés indépendant de la solution TA ; aussi d'autres solutions pourront en faire usage (*e.g.* coffre cryptographique).

La gestion du point 2 est assurée par le modèle de sécurité que nous définissons dans la prochaine sous-section.

3.3.2 Modèle de sécurité

Par les défenses logicielles proposées par l'IDE et de whiteboxes adaptées (Figure 2.6), nous pouvons nous concentrer sur la description du modèle en boîte noire du module FAST. Ce modèle s'attachera à quantifier l'information qu'un adversaire pourrait récolter

par différents enrôlements d'un même utilisateur. Alors que classiquement le modèle en boîte noire confère à un attaquant la vue des entrées/sorties d'un algorithme, ce ne sera pas le cas pour FAST puisque l'entrée (profil utilisateur Ω) est exactement la donnée à protéger en confidentialité. Notons le parallèle avec les Fuzzy Extractors où les modèles définis par Boyen [85] et Canetti *et al.* [94] ne confèrent pas non plus à l'adversaire l'accès au secret bruité w, qui là aussi, constitue le secret à protéger.

Ainsi, un utilisateur donné $\mathcal{U}$ devra pouvoir enrôler ou ré-enrôler plusieurs profils compatibles $\Omega^1, \ldots, \Omega^\rho$ sur différents serveurs sans que le cumul des données de reconstruction associées $\mathsf{hd}^1, \ldots, \mathsf{hd}^\rho$ ne permette à un adversaire d'en déduire de l'information significative.

Nous rappelons alors, par la Figure 3.1, la vue de haut niveau du modèle de sécurité concernant le module FAST .

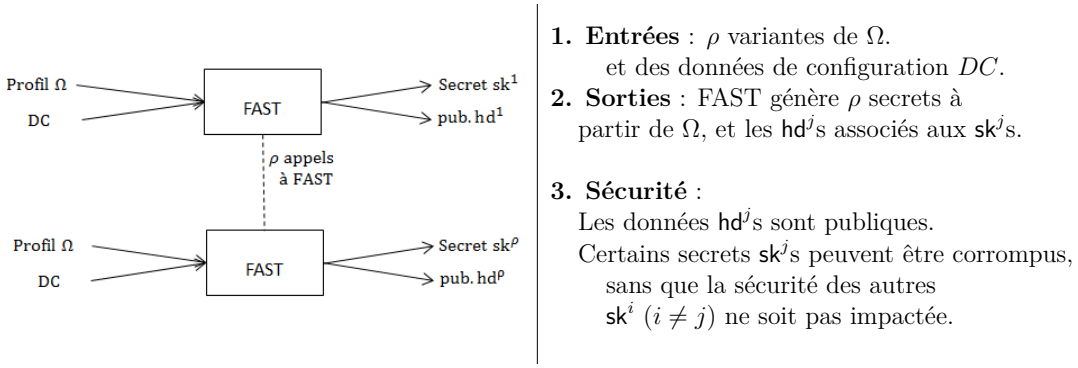

Figure 3.1: Aperçu du modèle de sécurité de FAST

Nous nous appuyons sur la réutilisabilité des FEs selon Canetti *et al.* [94] pour définir la réutilisabilité de FAST par les jeux décrits à la Figure 3.2. Considérons alors un challengeur $\mathcal{C}$ mettant au défi un adversaire $\mathcal{A}$ qui spécifiera ρ variables aléatoires $W^1, \ldots, W^\rho$ définies sur $\mathcal{ATT}^*$.

D'autre part, au cours du jeu, $\mathcal{A}$ pourra faire faire appel à l'oracle $\mathcal{O}^{\text{corrupt}}(.)$ pour lequel un nombre polynomial de q requêtes lui est accordé :

- étant donné un exposant j spécifié par l'adversaire, l'oracle lui retourne le secret associé sk^j. Le système est mis à jour : (j, sk^j) est ajouté à la liste des secrets corrompus SC.

Le challengeur $\mathcal{C}$ et l'adversaire $\mathcal{A}$ vont alors échanger selon le scénario décrit ci-dessous :

1. Le challengeur $\mathcal{C}$ exécute honnêtement le protocole $\mathsf{GenParam}$ pour obtenir les paramètres publics param dont la politique d'authentification PA.

2. L'adversaire $\mathcal{A}$ spécifie ρ variables aléatoires $W^1, \ldots, W^\rho$ définies sur $\mathcal{ATT}^*$ pour lesquelles les profils seront adaptés à la politique PA.
3. Pour $j = 1 \ldots, \rho$, le challengeur échantillonne $\Omega^j \xleftarrow{\$} W^j$ et calcule $(\mathsf{sk}^j, \mathsf{hd}^j) \leftarrow \mathsf{GenKey}(\Omega^j)$.
4. L'adversaire fait $q' \leqslant q$ requêtes à l'oracle $\mathcal{O}^{\text{corrupt}}$ puis choisit d'attaquer l'exposant j_0.
5. Dans le cas décisonnel, le challengeur retourne sk^{j_0} à l'adversaire si $b = 1$ et une valeur aléatoire $\mu \xleftarrow{\$} \{0,\}^l$ sinon.
6. L'adversaire peut encore faire $q - q'$ requêtes à l'oracle $\mathcal{O}^{\text{corrupt}}$.
7. Dans la version calculatoire, $\mathcal{A}$ termine le jeu en retournant une valeur $g \in \{0,1\}^l$ et gagne si $g = \mathsf{sk}^{j_0}$.
8. Dans la version décisionnelle, l'adversaire produit b^* et gagne si $b^* = b$.

Discussion Supposer que l'adversaire serait celui qui a le moins de connaissances sur les distributions des profils utilisateurs est injustifié et c'est en ce sens que la spécification des W^js lui est attribuée à l'étape 2. En effet, toute information accessible à un utilisateur honnête est potentiellement à disposition de l'adversaire. D'autre part, supposer ces distributions de profils confidentielles aux yeux de l'adversaire peut s'avérer dangereux : un sytème pourrait alors baser sa sécurité sur cette méconnaissance de l'adversaire mais s'il advenait, par quelque manière que ce soit, que ce dernier apprenne finalement ces certaines informations, cela remettrait en cause la sécurité globale du système.

Dans la version calculatoire du jeu, l'adversaire choisit un exposant à attaquer et essaie de deviner la clé secrète associée. La version décisionnelle propose une étape supplémentaire où l'adversaire doit décider si la valeur retournée par le challengeur est effectivement le secret associé sk^{j_0} ou une valeur aléatoirement tirée. Puisqu'un adversaire capable de retrouver le secret sk^{j_0} (version calculatoire du jeu) sera capable de le distinguer d'une valeur aléatoire, un module FAST prouvé sûr vis-à-vis de la version décisionnelle le sera aussi dans le cas calculatoire.

Nous formalisons le jeu, dans ses versions calculatoire et décisionnelle, à travers la Figure 3.2.

Définition 50 (Sécurité de FAST)**.** *Soit un $\mathcal{A}$ avec un nombre de requêtes polynomial $q(\lambda)$ à l'oracle $\mathcal{O}^{corrupt}$. On dira que le module FAST est q-réutilisable si les avantages $Adv^{calc}_{\mathcal{C},\mathcal{A}}(\lambda)$ et $Adv^{dec}_{\mathcal{C},\mathcal{A}}(\lambda)$ (Figure 3.2) sont négligeables.*

Dans la prochaine section, nous proposons l'usage de FEs pour instancier la définition d'un module FAST.

(a) Version calculatoire $\mathrm{Calc}_{\mathcal{C},\mathcal{A}}(\lambda)$

1. $\mathcal{C}$ génère $\mathsf{param} \leftarrow \mathsf{GenParam}(1^\lambda)$.
   avec $\mathsf{param} = ((\epsilon, s, l), \mathsf{PA}, \mathsf{DC})$.
2. Initialiser $\mathsf{SC} = \mathsf{L} = \emptyset$.
3. $\mathcal{A}$ spécifie les distributions $W^1, \ldots, W^\rho$.
4. Pour $j = 1 \ldots \rho$,
   $\mathcal{C}$ échantillone $\Omega^j \leftarrow W^j$.
   $(\mathsf{sk}^j, \mathsf{hd}^j) \leftarrow \mathsf{GenKey}(\Omega^j, \mathsf{PA}, \mathsf{DC})$, $\mathsf{L} \cup \mathsf{hd}^j$.
5. $(j_0, r) \leftarrow \mathcal{A}(\mathsf{param}, \mathsf{L}, \mathsf{SC} : \mathcal{O}^{\text{corrupt}})$.
6. Si $(j_0, \mathsf{sk}^{j_0}) \in \mathsf{SC}$, recommencer le jeu.
7. $\mathcal{A}$ gagne si $r = \mathsf{sk}^{j_0}$.

$$\mathrm{Adv}^{\text{calc}}_{\mathcal{C},\mathcal{A}}(\lambda) = |\Pr[r = sk^{j_0}] - 2^{-l}|$$

(b) Version décisionnelle $\mathrm{Dec}^b_{\mathcal{C},\mathcal{A}}(\lambda)$

Etapes 1,2 et 3 identiques à $\mathrm{Calc}_{\mathcal{C},\mathcal{A}}(\lambda)$.
3. $\mathcal{A}$ spécifie les distributions $W^1, \ldots, W^\rho$.
4. Pour $j = 1 \ldots \rho$,
   $\mathcal{C}$ échantillone $\Omega^j \leftarrow W^j$.
   $(\mathsf{sk}^j, \mathsf{hd}^j) \leftarrow \mathsf{GenKey}(\Omega^j, \mathsf{PA}, \mathsf{DC})$, $\mathsf{L} \cup \mathsf{hd}^j$.
5. $j_0 \leftarrow \mathcal{A}(\mathsf{param}, \mathsf{L}, \mathsf{SC} : \mathcal{O}^{\text{corrupt}})$.
   Si $b = 1$, $\mathcal{C}$ retourne $r = \mathsf{sk}^{j_0}$ à $\mathcal{A}$.
   Si $b = 0$, $\mathcal{C}$ retourne $r \xleftarrow{\$} \{0,1\}^l$ à $\mathcal{A}$.
6. $b^* \leftarrow \mathcal{A}(\mathsf{param}, \mathsf{L}, \mathsf{SC}, r : \mathcal{O}^{\text{corrupt}})$.
7. Si $(j_0, \mathsf{sk}^{j_0}) \in \mathsf{SC}$, recommencer le jeu.
   Sinon l'adversaire gagne si $b^* = b$.

$$\mathrm{Adv}^{\text{dec}}_{\mathcal{C},\mathcal{A}}(\lambda) = |\Pr[b^* = b] - \frac{1}{2}|$$

Figure 3.2: Sécurité de FAST

3.4 Instanciation de FAST et réponse aux limitations L2 et L3

Remarquons que la génération de clés au stade TA1 (Figures 2.3 et 2.4) semble satisfaire aux définitions relatives au module FAST (Définitions 49 et 50) nouvellement introduites. Cependant les différentes limitations évoquées nous ont conduit à la formalisation introduite ci-dessus et à l'instanciation que nous décrivons dans cette section.

3.4.1 Une solution basée sur les Fuzzy Extractors

Leur introduction au Chapitre 1 permet de souligner que les Fuzzy Extractors sont naturellement définis pour répondre à la limitation L2 (gestion d'attributs variables) tout en assurant les requis R5, R6, R7 et R8 (*respectivement* la génération de secret, le respect de la vie privée, l'exploitation des sources d'aléa la gestion de secret). Par définition, un FE permettra d'associer un secret stable à tout attribut variable. Ainsi, les FEs permettent naturellement la gestion du niveau 1 de la politique de sécurité et nous verrons également qu'un FE dédié permettra de s'assurer de la réalisation du niveau 2.

Aperçu et idée générale

Par un enchaînement de trois figures, nous présenterons par zooms successifs le fonctionnement de notre instanciation. Le but final est de générer un secret cryptographique (primitive GenKey) à partir d'un profil utilisateur Ω en respectant une politique d'authentification PA décidée par le serveur d'authentification à partir d'un niveau de sécurité λ (primitive $\mathsf{GenParam}$). D'autre part, nous avons vu que ce secret ne

peut pas être stocké de manière sécurisée et doit donc être reconstruit à chaque phase d'authentification : c'est le rôle de la primitive RepKey. Pour ce faire, Une donnée publique pourra éventuellement être générée par GenKey afin d'aider aux reconstructions futures.

Reprenant la représentation du SDSK B&W (Figure 2.6), le positionnement de FAST est précisé par la Figure 3.3.

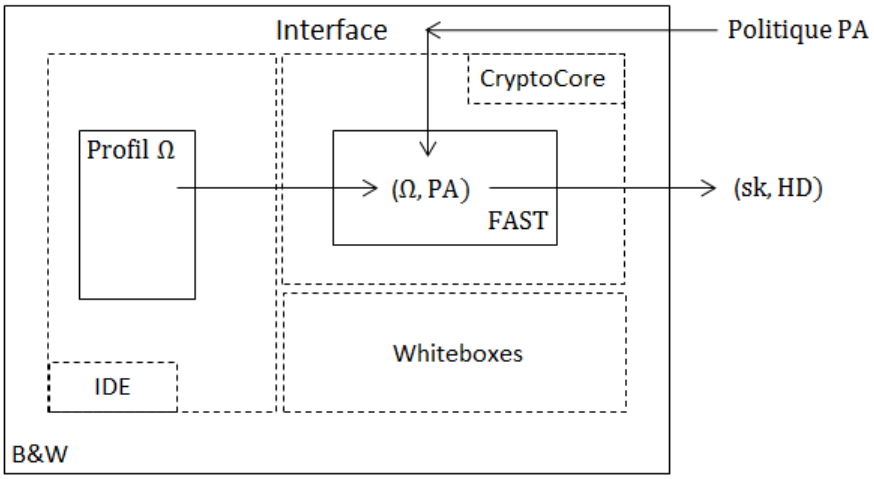

Figure 3.3: Fonctionnement global de FAST

Par usage d'un FE (*resp.* d'un extracteur) adapté, chaque attribut variable (*resp.* statique) se verra attribué une donnée stable et aléatoire définie comme *secret partiel*. Par définition de la primitive associée (FE ou extracteur), chaque secret partiel sera retrouvé lors d'une phase de reconstruction (RepKey) si l'attribut correspondant respecte le niveau 1 de la politique d'authentification. La figure 3.4 reprend cette idée.

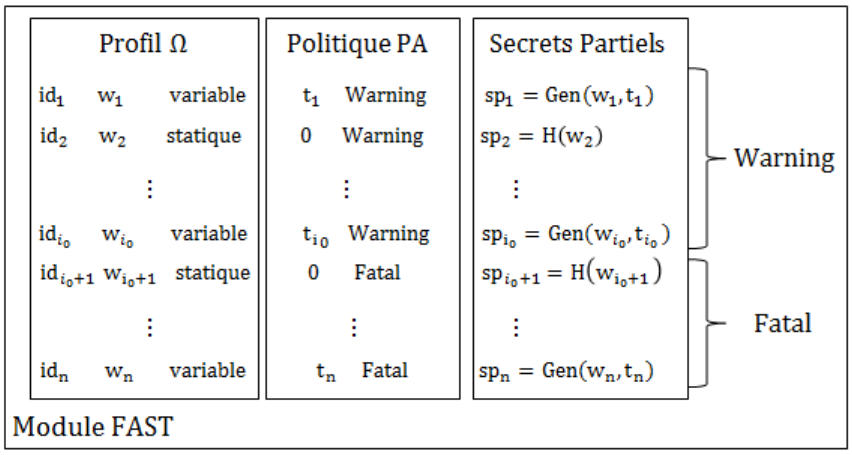

Figure 3.4: Stabilisation des attributs par des secrets partiels. H et Gen désignent respectivement un extracteur et un FE génériques.

Cette stabilisation des attributs par l'usage des secrets partiels permettra alors la

gestion du niveau 2 de la politique d'authentification. L'idée est de générer un secret global associé aux attributs estampillés *Fatal* et, par l'usage d'un FE adapté, de générer également un secret global associé aux attributs estampillés *Warning*. Ce FE aura pour tolérance le nombre de warning stipulés t_{n+1} par la politique d'authentification PA. Finalement, comme dépeint à la Figure 3.5, le secret sk sera issus de ces deux secrets.

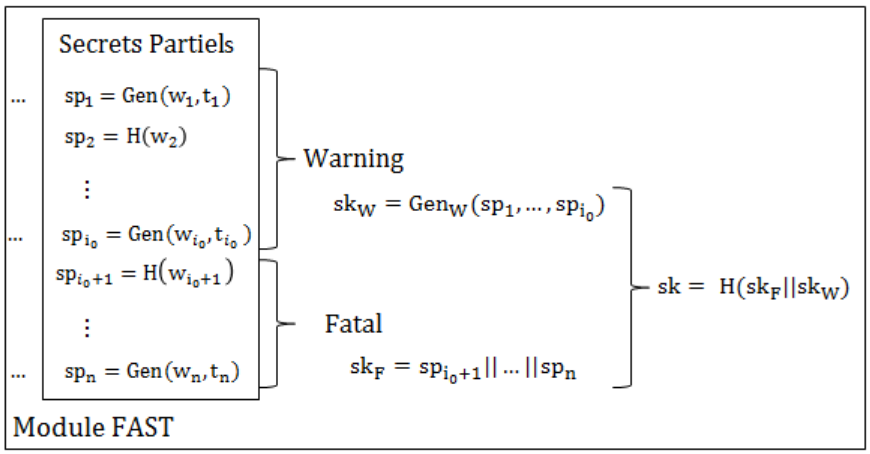

Figure 3.5: Gestion du niveau 2 et génération du secret sk via les secrets partiels

3.4.2 Outils et Hypothèses

Outils mis en jeu Extracteurs forts et Fuzzy Extractors permettent d'extraire l'entropie contenue dans une source (voir Chapitre 1 pour plus de détails). En particuliers, les Fuzzy Extractors peuvent se voir comme des extracteurs forts pour lesquels l'entrée pourra subir un certain nombre d'erreurs tout en étant capable d'extraire le même aléa lors des phases de reconstruction. Ces deux primitives largement étudiés dans la littérature [34, 35, 36, 37, 38] et [12, 85, 13, 86, 83, 94] jouissent de définitions statistique et calculatoire permettant de quantifier la proximité entre leurs sorties et l'uniforme. Le module FAST est défini du point de vue calculatoire (Définiton 49, point 4.) car 1) ce modèle couvre une sécurité dans le monde réel et 2) notre instantiation s'appuiera sur des constructions, elles-mêmes prouvées sures dans ce modèles [94, 5].

D'autre part, le modèle de sécurité de FAST sous-entend une réutilisabilité des FEs sous-jacents ; ainsi, nous ne pourrons pas utiliser les constructions proposées Ainsi, la sécurité de notre instanciation sera prouvée dans le modèle de l'oracle aléatoire et fera appel aux outils suivants :

1. $\mathsf{H}_i : \{0,1\}^{n_i} \times \{0,1\}^{r_i} \to \{0,1\}^{l_i}$ est une fonction de hachage cryptographique modélisée comme un oracle aléatoire. Reprenant la remarque 1, H_i fera office d'extracteur et sera de sécurité (s, ϵ) dans le sens où il vérifiera

$\delta^{\mathcal{D}_s}((\mathsf{H}_i(W;X),X),(U_{l_i},X)) \leqslant \epsilon$, pour toute distribution de probabilités W sur $\{0,1\}^n$ et X est uniforme sur $\{0,1\}^{r_i}$. En pratique, les H_i permettront alors la génération de secrets cryptographiques à partir d'attributs statiques;

2. $(\mathsf{Gen}_i, \mathsf{Rep}_i)$ est un $(\mathcal{M}_{\mathsf{id}_i}, \mathcal{W}_i, l_i, t_i)$-FE de sécurité (ρ, s, ϵ) où $\mathcal{W}_i$ est une famille de distribution de probabilités sur $\mathcal{M}_{\mathsf{id}_i}$.

3. $(\mathsf{Gen}_{n+1}, \mathsf{Rep}_{n+1})$ est un $(\mathcal{M}_{n+1}, \mathcal{W}_{n+1}, l_{n+1}, t_{n+1})$-FE dans le cas moyen de sécurité (ρ, s, ϵ) où $\mathcal{W}_{n+1}$ est une famille de distribution de probabilités sur $\mathcal{M}_{n+1} \stackrel{\text{def}}{=} \mathsf{SDif}_{|\mathsf{LW}|}(\{0,1\}^{nw})$.

À des fins rédactionnelles, nous explicitons la graine issue de X définissant l'aléa mis en jeu par H_i ; en revanche, reprenant les notations usuelles [13, 83], la graine d'aléa utilisée par un FE n'apparaîtra pas explicitement et sera supposé stockée avec la valeur publique P_i retournée par Gen_i.

La description proposée dans la suite reposera donc sur fonctions de hachage et FEs réutilisables, excluant alors les constructions de Dodis *et al.* [13]. Le seul FE réutilisable, à l'époque des travaux, est celui dû à R. Canetti, B. Fuller, O. Paneth, L. Reyzin et A. D. Smith [94]. Cependant et bien que basé sur la métrique de Hamming, leur construction pourra, *sous conditions*, s'étendre à la différence d'ensembles via l'équivalence bin-set. En effet, dans ce cas, on devra supposer que les éléments de l'ensemble w proviendront alors d'un univers de petit cardinal *i.e.* polynomial en la taille de w : c'est le scénario du *petit univers* (ou *small universe setting*). Comme nous l'avons vu, cette métrique est la plus adaptée à notre contexte mais la Section 3.6 permettra également d'exploiter la distance de Hamming en adaptant le traitement de nos attributs binaires.

Hypothèses Avant de formuler deux hypothèses sur lesquelles notre instanciation va s'appuyer, nous proposons les notations suivantes.

Notations Les notations présentées ci-dessous seront conservées jusqu'à la fin de ce chapitre.
Soit $\mathcal{W}$ une famille de distributions de probabilités définies sur $\mathcal{ATT}^n$. Soit une distribution $W \in \mathcal{W}$, $\Omega \xleftarrow{\$} W$ signifie que l'événement Ω a été aléatoirement tiré de W. Maintenant, en écrivant $\Omega = (\mathsf{att}_1, \ldots, \mathsf{att}_n)$, où pour tout i, $\mathsf{att}_i = (\mathsf{id}_i, w_i, b_i)$, on note $\mathcal{W}_i$ une famille de distribution de probabilités définie sur $\mathcal{M}_{\mathsf{id}_i}$.

Nous énonçons alors les hypothèses suivantes :

Hyp1. $\Omega = (\mathsf{att}_1, \ldots, \mathsf{att}_n) \xleftarrow{\$} W \Leftrightarrow \forall i = 1 \ldots n, \exists WA_i \in \mathcal{W}_i, (w_1, \ldots, w_n) \xleftarrow{\$} WA_1 \times \ldots WA_n$.

Hyp2. $\forall 1 \leqslant i \neq j \leqslant n$, WA_i et WA_j sont indépendantes.

L'hypothèse **Hyp1** souligne qu'un profil utilisateur Ω est caractérisé par ses valeurs d'attributs $(w_1, \ldots, w_n)$. Comme mentionné lors de la description de la réutilisabilité (Sous-section 3.3.2), l'adversaire a accès aux distributions de probabilités dont sont issus les profils. Ainsi, à des fins d'opérabilité et de sécurité, les identifiants d'attributs – ainsi que leur nature statique ($b_i = 0$) ou variable ($b_i = 1$)– sont connus de l'adversaire. La seule information secrète aux yeux de l'adversaire est la valeur prise par chacun des attributs.

L'hypothèse **Hyp2** suppose que les différents attributs constituant un profil utilisateur sont indépendants deux à deux[1].

3.4.3 Description formelle

Notations, outils et hypothèses précédemment définis vont nous permettre de présenter notre instanciation de la Définition 49. Avant toute chose, nous apportons une clarification sur les notations.

Précisions informelles Dans toute la suite, la donnée de reconstruction hd associée au profil Ω sera un vecteur s'écrivant $\mathsf{hd} = (\mathsf{hd}_1, \ldots, \mathsf{hd}_n, \mathsf{hd}_{n+1}, \mathsf{salt})$ où l'indice $i \leqslant n$ sera associé au i$^{\text{ième}}$ attribut de Ω et salt une graine aléatoire. L'hypothèse **Hyp1** permet alors d'associer chaque hd_i à w_i $(1 \leqslant i \leqslant n)$.

Génération des paramètres par GenParam

La primitive $\mathsf{GenParam}$ prend en entrée le paramètre de sécurité λ pour retourner des paramètres $\mathsf{keyParam}(\lambda) = (\epsilon, s, l)$, une politique d'authentification $\mathsf{PA}(\lambda)$, et des données de configuration $\mathsf{DC}(\lambda)$. Dans le but d'alléger la notation, nous omettons le terme λ. Précisons le rôle de chacun des termes :

1. **Clé FAST.** Le terme $\mathsf{keyParam} = (\epsilon, s, l)$ permet de quantifier la distance entre la clé retournée par FAST sk et la distribution uniforme U_l selon la propriété de sécurité du module FAST (Définition 49) ;

2. **Politique d'authentification.** La politique d'authentification permettra de sélectionner des attributs contenant assez d'entropie pour constituer un profil utilisateur adapté au niveau de sécurité.

3. **Données de Configuration.** Dans notre instanciation, elles permettront de préciser les outils à utiliser pour appliquer la politique d'authentification. Plus précisément, DC spécifie quel type d'extracteur permettra de générer le secret partiel associé à un attribut.

[1] Nous verrons que les résultats proposés au Chapitre 4, proposés dans l'optique de répondre à la réutilisabilité, nous permettront également de nous affranchir de cette hypothèse

Nous décrivons alors cette primitive par la Figure 3.6.

Entrée : 1^λ.

1. Initialisation de $\mathsf{keyParam}$.
   $(r, s, \epsilon) \in \mathbb{N}^2 \times \mathbb{R}$.

2. Génération de $\mathsf{PA} = (\mathsf{Id}, t, b^2) \in \Gamma$.
   a. $\mathsf{Id} = (\mathsf{id}_1, \ldots, \mathsf{id}_n)$,
   b. $t = (t_1, \ldots, t_n, t_{n+1})$,
   c. $b^2 = (b_1^2, \ldots, b_n^2)$.

3. Génération de $\mathsf{DC} = (\mathsf{Ext}_1, \ldots, \mathsf{Ext}_{n+1})$:
   a. si id_i identifiant d'attribut variable,
      $\mathsf{Ext}_i = (\mathsf{Gen}_i, \mathsf{Rep}_i)$ est un $(\mathcal{M}_{\mathsf{id}_i}, \mathcal{W}_i, l_i, t_i)$-FE de sécurité (s, ϵ).
   b. sinon, $\mathsf{Ext}_i = \mathsf{H}_i : \{0,1\}^{n_i} \times \{0,1\}^{r_i} \to \{0,1\}^{l_i}$ de sécurité (s, ϵ).
   c. $\mathsf{Ext}_{n+1} = (\mathsf{Gen}_{n+1}, \mathsf{Rep}_{n+1})$ est
      un $(\mathcal{M}_{n+1}, \mathcal{W}_{n+1}, l_{n+1}, t_{n+1})$-FE de sécurité (s, ϵ).

4. Finalisation de $\mathsf{keyParam}$.
   a. $l = \sum_{i:b_i^2=0} l_i + l_{n+1}$,
   b. $\mathsf{keyParam} = (r, s, l, \epsilon)$.

Retourner $(\mathsf{keyParam}, \mathsf{PA}, \mathsf{DC})$.

Figure 3.6: Primitive GenParam

Génération de clés par GenKey

La primitive GenKey prend en entrées un profil utilisateur $\Omega = (\mathsf{att}_1, \ldots, \mathsf{att}_n)$ ainsi que la politique d'authentification PA et les données de configuration DC générées par GenParam. Puisqu'un profil est en fait un vecteur d'attributs de longueur n, chaque attribut s'écrit $\mathsf{att}_i = (\mathrm{id}_i, w_i, b_i)$ où id_i désigne l'identifiant, w_i sa valeur et b_i sa variabilité. On initialise $\mathsf{sk}_F = ""$ et $w_W = \emptyset$. Dans un premier temps, chaque attribut est traité individuellement selon le niveau 1 de la politique de d'authentification puis, la génération du secret FAST sk sera assurée par le niveau 2.

1. **Niveau 1.** Génération des secrets partiels selon les cas :

   - *statique et fatal.* $\mathsf{DC}[i]$ spécifie alors l'extracteur $\mathsf{H}_i : \{0,1\}^{n_i} \to \{0,1\}^{l_i}$ de sécurité (s, ϵ) qui retourne le secret partiel $\mathsf{sp}_i \leftarrow \mathsf{H}_i(w_i; \mathsf{hd}_i)$, où $\mathsf{hd}_i \xleftarrow{\$} \{0,1\}^{r_i}$ est aléatoirement choisie. Le secret de type fatal sk_F est mis à jour $\mathsf{sk}_F = \mathsf{sk}_F \| \mathsf{sp}_i$;
   - *variable et fatal.* $\mathsf{DC}[i]$ spécifie $(\mathsf{Gen}_i, \mathsf{Rep}_i)$ le $(\mathcal{M}_{\mathsf{id}_i}, \mathcal{W}_i, l_i, t_i)$-FE de sécurité (s, ϵ) pour gérer les variations de l'attribut. On obtient alors secret partiel

et donnée publique comme suit $(\mathsf{sp}_i, \mathsf{hd}_i) \leftarrow \mathsf{Gen}_i(w_i)$. sk_F est mis à jour : $\mathsf{sk}_F = \mathsf{sk}_F \| \mathsf{sp}_i$;

- *statique et warning.* $\mathsf{DC}[i]$ permet de générer $\mathsf{sp}_i \leftarrow \mathsf{H}_i(w_i; \mathsf{hd}_i)$ où $\mathsf{hd}_i \xleftarrow{\$} \{0,1\}^{r_i}$. w_W est mis à jour $w_W = w_W \cup \mathsf{sp}_i$;
- *variable et warning.* $\mathsf{DC}[i]$ permet de générer $(\mathsf{sp}_i, \mathsf{hd}_i) \leftarrow \mathsf{Gen}_i(w_i)$. w_W est mis à jour : $w_W = w_W \cup \mathsf{sp}_i$.

2. **Niveau 2.** Gestion du niveau 2 :

- Le secret de type *warning* est généré comme suit $(\mathsf{sk}_W, \mathsf{hd}_{n+1}) \leftarrow \mathsf{Gen}_{n+1}(w_W, t_{n+1})$, où t_{n+1} est le nombre de warning autorisés par PA ;
- le secret global sk est généré à partir de sk_F et sk_W : $\mathsf{sk} \leftarrow \mathcal{H}_l(\mathsf{sk}_F \| \mathsf{sk}_W; \mathsf{salt})$, avec salt une graine aléatoire.
- le vecteur $\mathsf{hd} \stackrel{\text{def}}{=} (\mathsf{hd}_1, \ldots, \mathsf{hd}_n, \mathsf{hd}_{n+1}, salt)$.

La Figure 3.7 formalise la primitive GenKey.

Entrées : $\Omega, \mathsf{PA}, \mathsf{DC}$.

1. Initialisation
$\mathsf{keyParam} == (r, s, l, \epsilon)$.
$\mathsf{PA} = (\mathsf{Id}, t, b^2) \in \Gamma$,
$\mathsf{DC} = (\mathsf{Ext}_1, \ldots, \mathsf{Ext}_{n+1})$,
$\Omega = (\mathsf{att}_1, \ldots, \mathsf{att}_n) \in \mathcal{ATT}^n$, avec $\mathsf{att}_i = (\mathsf{Id}_i, w_i, b_i)$,
$\mathsf{sk}_F = ""$; $w_W = \emptyset$.

2. Traitement
Pour $i = 1 \ldots n$:
a. Gestion du niveau 1
Si $b_i = 0$, $\mathsf{DC}[i] = \mathsf{H}_i : \{0,1\}^{n_i} \times \{0,1\}^{r_i} \to \{0,1\}^{l_i}$,
$\mathsf{hd}_i \xleftarrow{\$} \{0,1\}^{r_i}$,
$\mathsf{sp}_i \leftarrow \mathsf{H}_i(w_i; \mathsf{hd}_i)$.
Sinon $(b_i = 1)$, $\mathsf{DC}[i] = (\mathsf{Gen}_i, \mathsf{Rep}_i)$,
$(\mathsf{sp}_i, \mathsf{hd}_i) \leftarrow \mathsf{Gen}_i(w_i)$.

b. Gestion du niveau 2
Si $b_i^2 = 0$, $\mathsf{sk}_F = \mathsf{sk}_F \| \mathsf{sp}_i$,
Sinon $b_i^2 = 1$,
Masquer $\mathsf{sp}_i \leftarrow \mathsf{sp}_i \| \mathsf{pad}_i \in \{0,1\}^{n_W}$,
$w_W = w_W \cup \mathsf{sp}_i$.

3. Génération de hd et sk
$\mathsf{DC}[n+1] = (\mathsf{Gen}_{n+1}, \mathsf{Rep}_{n+1})$,
a. $(\mathsf{sk}_W, \mathsf{hd}_{n+1}) = \mathsf{Gen}_{n+1}(w_W)$,
b. $\mathsf{salt} \xleftarrow{\$} \{0,1\}^r$,
$\mathsf{sk} \leftarrow \mathsf{H}_l(\mathsf{sk}_F \| \mathsf{sk}_W; \mathsf{salt})$,
c. $\mathsf{hd} \stackrel{\text{def}}{=} (\mathsf{hd}_1, \ldots, \mathsf{hd}_n, \mathsf{hd}_{n+1}, \mathsf{salt})$.

Retourner $(\mathsf{sk}, \mathsf{hd})$.

Figure 3.7: Primitive GenKey

Reproduction de clés par RepKey

La primitive RepKey, formellement décrite par la Figure 3.8 prend en entrées un profil utilisateur Ω', la politique d'authentification PA, les données de configuration DC, et la donnée de reconstruction $\mathsf{hd} = (\mathsf{hd}_1, \ldots, \mathsf{hd}_n, \mathsf{hd}_{n+1}, \mathsf{salt})$ générée par GenKey. $\Omega' = (\mathsf{att}'_1, \ldots, \mathsf{att}'_n)$ présentera des attributs dont les valeurs w'_i pourront potentiellement différer des w_i du profil d'enrôlement Ω. De manière analogue à GenKey, $sk'_F = ""$

et $w'_W = \emptyset$ sont initialisées vides. En rappelant que les données de configuration DC spécifient quels extracteurs doivent être appliquer à chaque w'_i, le déroulement de RepKey a lieu comme suit:

1. **Niveau 1.** Re-génération des secrets partiels :
   - *statique et fatal.* $\mathsf{DC}[i]$ spécifie H_i pour générer $\mathsf{sp}'_i \leftarrow \mathsf{H}_i(w'_i; \mathsf{hd}_i)$. sk'_F est mis à jour $\mathsf{sk}'_F = \mathsf{sk}'_F \| \mathsf{sp}'_i$;
   - *variable et fatal.* $\mathsf{DC}[i]$ spécifie le FE pour calculer $\mathsf{sp}'_i \leftarrow \mathsf{Rep}_i(w'_i, \mathsf{hd}_i)$. sk'_F est mis à jour $\mathsf{sk}'_F = \mathsf{sk}'_F \| \mathsf{sp}'_i$;
   - *statique et warning.* $\mathsf{sp}'_i \leftarrow \mathsf{H}_i(w'_i; \mathsf{hd}_i)$. Le secret de type warning w'_W est mis à jour $w'_W = w'_W \cup \mathsf{sp}'_i$;
   - *variable et warning.* $\mathsf{sp}'_i \leftarrow \mathsf{Rep}_i(w'_i, \mathsf{hd}_i)$. w'_W est mis à jour $w'_W = w'_W \cup \mathsf{sp'}_i$.
2. **(Niveau 2)** Gestion du niveau 2 :
   - Parser $\mathsf{DC}[n+1] = (\mathsf{Gen}_{n+1}, \mathsf{Rep}_{n+1})$;
   - Le secret de type warning est (re-)généré comme suit $\mathsf{sk}'_W \leftarrow \mathsf{Rep}_{n+1}(w'_W, \mathsf{hd}_{n+1})$;
   - Identiquement à la génération GenKey, on génère $\mathsf{sk}' \leftarrow \mathsf{H}_l(\mathsf{sk}'_F \| \mathsf{sk}'_W; \mathsf{salt})$.

La reconstruction de clé via RepKey est décrite en Figure 3.8.

Entrées : $\Omega', \mathsf{PA}, \mathsf{DC}, \mathsf{hd}$.

1. Initialisation
$\Omega' = (\mathsf{att}'_1, \ldots, \mathsf{att}'_n) \in \mathcal{ATT}^n$,
avec $\mathsf{att}'_i = (\mathsf{Id}_i, w'_i, b_i)$,
$\mathsf{PA} = (\mathsf{Id}, t, b^2) \in -$,
$\mathsf{DC} = (\mathsf{Ext}_1, \ldots, \mathsf{Ext}_{n+1})$,
$\mathsf{hd} = (\mathsf{hd}_1, \ldots, \mathsf{hd}_n, \mathsf{hd}_{n+1}, \mathsf{salt})$,
$\mathsf{sk}'_F = ""$; $w'_W = \emptyset$.

2. Traitement
Pour $i = 1 \ldots n$:
a. Gestion du niveau 1
Si $b_i = 0$, $\mathsf{Ext}_i = \mathsf{H}_i : \{0,1\}^{n_i} \times \{0,1\}^{r_i} \to \{0,1\}^{l_i}$,
$\mathsf{sp}'_i \leftarrow \mathsf{H}_i(w'_i; \mathsf{hd}_i)$.
Sinon ($b_i = 1$), $\mathsf{Ext}_i = (\mathsf{Gen}_i, \mathsf{Rep}_i)$,
$\mathsf{sp}'_i \leftarrow \mathsf{Rep}_i(w'_i, \mathsf{hd}_i)$.

b. Gestion du niveau 2
Si $b_i^2 = 0$, $\mathsf{sk}'_F = \mathsf{sk}'_F \| \mathsf{sp}'_i$.
Sinon $b_i^2 = 1$,
Masquer $\mathsf{sp}'_i \leftarrow \mathsf{sp}'_i \| \mathsf{pad}_i \in \{0,1\}^{n_W}$,
$w'_W = w'_W \cup \mathsf{sp}'_i$.

3. (Re-)génération de la clé FAST
a. $\mathsf{DC} = (\mathsf{Gen}_{n+1}, \mathsf{Rep}_{n+1})$,
b. $\mathsf{sk}'_W \leftarrow \mathsf{Rep}_{n+1}(w'_W, P_{n+1})$.
c. $\mathsf{sk} \leftarrow \mathsf{H}_l(\mathsf{sk}_F \| \mathsf{sk}_W; \mathsf{salt})$.

Retourner sk.

Figure 3.8: Primitive RepKey

3.4.4 Consistance et Sécurité

Théorème 1. *Les Figures 3.6, 3.7 et 3.8 définissent un module FAST satisfaisant la Définition 49.*

Proof. Nous commençons par la consistance du module ainsi instancié avant de nous concentrer sur la propriété de sécurité.

Correctness

Supposons que la primitive $\mathsf{GenParam}$ ait été exécutée correctement. En reprenant les notations, nous rappelons que la donnée $\mathsf{DC}[i]$ stipule quel extracteur utiliser pour calculer sp_i : à savoir H_i ou $(\mathsf{Gen}_i, \mathsf{Rep}_i)$ en fonction de la variabilité b_i de l'attribut att_i. Soient Ω et Ω' deux profils compatibles selon PA ($\Omega \approx_{\mathsf{PA}} \Omega'$) alors :

- pour tout identifiant d'attribut id_i *statique* et *fatal*, on aura $w_i = w'_i$ de sorte que $\mathsf{sp}_i = \mathsf{H}_i(w_i; \mathsf{hd}_i) = \mathsf{H}_i(w'_i; \mathsf{hd}_i)$;
- pour tout identifiant d'attribut id_i *variable* et *fatal*, on aura $d(w_i, w'_i) \leqslant t_i$ de sorte que si $(\mathsf{sp}_i, \mathsf{hd}_i) = \mathsf{Gen}_i(w_i; \mathsf{salt}_i)$ alors $\mathsf{sp}_i = \mathsf{sp}'_i = \mathsf{Rep}_i(w'_i, \mathsf{hd}_i)$;
- $\Omega \approx_{\mathsf{PA}} \Omega'$ donc moins de t_{n+1} attributs att_i estampillés *warning* ayant pour valeur w'_i conduiront à la reconstruction d'un secret partiel $\mathsf{sp}'_i \neq \mathsf{sp}_i$. Ainsi, le FE $(\mathsf{Gen}_{n+1}, \mathsf{Rep}_{n+1})$ corrigeant t_{n+1} erreurs pourra reconstruire sk_W avec succès.

Les secrets sk_F et sk_W ainsi reconstruits mènent trivialement à la re-génération attendue de sk via l'usage de H_l.

Sécurité

Désormais, nous prouvons la propriété de sécurité définie par le point 4. de la Définition 49. Informellement, il faut montrer que la clé générée sk est indistinguable de l'uniforme même en présence de hd.

Notation Par **Hyp1**, nous avons que pour tout $i = 1 \dots n$, $w_i \xleftarrow{\$} WA_i$. De manière analogue nous noterons, $(\mathsf{SP}_i, \mathsf{HD}_i) \leftarrow \mathsf{GenKey}(WA_i)$ les distributions dont sont issues les couples images $(\mathsf{sp}_i, \mathsf{hd}_i)$.

Par la génération d'une politique d'authentification PA, La primitive $\mathsf{GenParam}$ induit $\mathcal{W}_{\mathsf{PA}}$, une famille de distributions de probabilités adaptée à PA. Il existe alors $W \in \mathcal{W}_{\mathsf{PA}}$ telle que $\Omega \xleftarrow{\$} W$. Par **Hyp1**, $\forall i = 1 \dots n, \exists\, WA_i, (w_1, \dots, w_n) \xleftarrow{\$} WA_1 \times \dots WA_n$.

Ainsi pour tout attribut estampillé *warning* (*resp. fatal*), le couple $(\mathsf{sp}_i, \mathsf{hd}_i)$ est généré par le FE (*resp.* l'oracle) associé. Reprenant la notation tout juste introduite, $(\mathsf{SP}_i, \mathsf{HD}_i)$ est la distribution induite par $\mathsf{Gen}(WA_i)$ (*resp.* $\mathsf{H}_i(WA_i; U_{r_i})$. Dans les deux cas, la sécurité de l'extracteur sous-jacent assure que :

$$\delta^{\mathcal{D}_s}((\mathsf{SP}_i, \mathsf{HD}_i), (U_{l_i}, \mathsf{HD}_i)) \leqslant \epsilon. \tag{3.1}$$

D'après l'hypothèse **Hyp2**, les WA_is sont indépendantes deux à deux *i.e.* pour tout $i \neq j$, WA_i et WA_j sont deux distributions aléatoires indépendantes. Par conséquent :

$$\forall i \neq j, (\mathsf{SP}_i, \mathsf{HD}_i) \text{ et } (\mathsf{SP}_j, \mathsf{HD}_j) \text{ sont indépendantes.} \tag{3.2}$$

Pour tout $1 \leqslant i \leqslant n$, notons $X_i \stackrel{\text{def}}{=} (\mathsf{SP}_i, \mathsf{HD}_i)$ et $Y_i \stackrel{\text{def}}{=} (U_{l_i}, \mathsf{HD}_i)$ où $(\mathsf{SP}_i, \mathsf{HD}_i) \leftarrow \mathsf{Gen}_i(WA_i)$. Réécrire l'équation 3.2 avec cette notation revient à dire que les X_is sont indépendantes.

Nous rappelons alors le lemme suivant.

Lemme 1 ([147])**.** *Soient X et Y deux distributions calculatoirement indistinguables vérifiant $\delta^{\mathcal{D}_s}(X, Y) \leqslant \epsilon$. Soit f une fonction qui termine en temps polynomial alors $\delta^{\mathcal{D}_s}(f(X), f(Y)) \leqslant \epsilon$.*

Définissons alors la fonction f_{X_j} paramétrée par X_j telle que $f_{X_j}(Z) := (Z, X_j)$. Maintenant, puisque $X_i \approx_{s,\epsilon} Y_i$, on aura que $f_{X_j}(X_i) \approx_{s,\epsilon} f_{X_j}(Y_i)$. *i.e.* $(X_i, X_j) \approx_{s,\epsilon} (Y_i, X_j)$. Ce qui se réécrit sous la forme :

$$(\mathsf{SP}_i, \mathsf{HD}_i, \mathsf{SP}_j, \mathsf{HD}_j) \approx_{s,\epsilon} (U_{l_i}, \mathsf{HD}_i, \mathsf{SP}_j, \mathsf{HD}_j)$$

Finalement, en rappelant que SP_j et U_{l_j} sont indistinguables et en changeant l'odre des variables nous obtenons :

$$(\mathsf{SP}_i, \mathsf{SP}_j, \mathsf{HD}_i, \mathsf{HD}_j) \approx_{s,\epsilon} (U_{l_i}, U_{l_j}, \mathsf{HD}_i, \mathsf{HD}_j). \tag{3.3}$$

En généralisant le raisonnement à toutes les variables, nous aboutissons à l'approximation 3.4,

$$(\mathsf{SP}_1, \ldots, \mathsf{SP}_n, \mathsf{HD}_1, \ldots, \mathsf{HD}_n) \approx_{s,\epsilon} (U_{l_1}, \ldots, U_{l_n}, \mathsf{HD}_1, \ldots, \mathsf{HD}_n). \tag{3.4}$$

Nous allons désormais traiter séparément les attributs estampillés *warning* et ceux estampillés *fatal*. Notons $\mathsf{LF} \stackrel{\text{def}}{=} \{i_1, \ldots, i_f\}$ et $\mathsf{LW} \stackrel{\text{def}}{=} \{j_1, \ldots, j_w\}$ avec $[\![n]\!] = \mathsf{LF} \sqcup \mathsf{LW}$.

En se concentrant sur le cas *fatal* et en reprenant le raisonnement, nous aboutissons

à l'approximation 3.5 :

$$(\mathsf{SP}_{i_1}, \ldots, \mathsf{SP}_{i_f}, \mathsf{HD}_{i_1}, \ldots, \mathsf{HD}_{i_f}) \approx_{s,\epsilon} (U_{l_{i_1}}, \ldots, U_{l_{i_f}}, \mathsf{HD}_{i_1}, \ldots, \mathsf{HD}_{i_f}). \tag{3.5}$$

Reprenant l'étape 2.b. de l'algorithme 3.7, le secret associé s'écrit $\mathsf{sk}_F = \mathsf{sp}_{i_1} \| \mathsf{sp}_{i_2} \| \ldots \| \mathsf{sp}_{i_f}$. Nous alors la distribution associée $\mathsf{SK}_F \stackrel{\text{def}}{=} (\mathsf{SP}_{i_1}, \mathsf{SP}_{i_2}, \ldots, \mathsf{SP}_{i_f})$ et $\mathsf{HD}_F \stackrel{\text{def}}{=} (\mathsf{HD}_{i_1}, \mathsf{HD}_{i_2}, \ldots \mathsf{HD}_{i_f})$ pour obtenir :

$$(\mathsf{SK}_F, \mathsf{HD}_F) \approx_{s,\epsilon} (U_{l_{i_1}+\cdots+l_{i_f}}, \mathsf{HD}_F) \tag{3.6}$$

Dans le cas *warning*, le raisonnement est similaire et nous avons l'approximation :

$$(\mathsf{SP}_{j_1}, \ldots, \mathsf{SP}_{j_w}, \mathsf{HD}_{j_1}, \ldots, \mathsf{HD}_{j_w}) \approx_{s,\epsilon} (U_{l_{j_1}}, \ldots, U_{l_{j_w}}, \mathsf{HD}_{j_1}, \ldots, \mathsf{HD}_{j_w}). \tag{3.7}$$

Rappelons que le secret associé sk_W, généré à l'étape 3.a. de l'algorithme 3.7 vérifie :

$$(\mathsf{sk}_W, \mathsf{HD}_{n+1}) \approx_{s,\epsilon} (U_{l_{n+1}}, \mathsf{HD}_{n+1}), \text{où} \leftarrow (\mathsf{sk}_W, \mathsf{HD}_{n+1}) \leftarrow \mathsf{Gen}_{n+1}(w_W)$$

Dans ce cas, les hd_is ($i \in \mathsf{LW}$), constituent de l'information additionnelle sur w_W à laquelle un attaquant aura accès. Notons alors I la distribution associée $I \stackrel{\text{def}}{=} (\mathsf{HD}_{j_1}, \ldots, \mathsf{HD}_{j_w})$ et SK_W celle dont sont issues les sk_Ws.

Puisque $(\mathsf{Gen}_{n+1}, \mathsf{Rep}_{n+1})$ est un FE dans le cas moyen, on obtient :

$$(\mathsf{SK}_W, \mathsf{HD}_{n+1}, I) \approx_{s,\epsilon} (U_{l_{n+1}}, \mathsf{HD}_{n+1}, I). \tag{3.8}$$

D'une part, nous avons que $\mathsf{HD} = (\mathsf{HD}_F, \mathsf{HD}_{n+1}, I)$, éventuellement en remettant les indices dans l'ordre ; d'autre part, puisque $(\mathsf{SK}_F, \mathsf{HD}_F)$ et $(\mathsf{SK}_W, \mathsf{HD}_{n+1}, I)$ sont des distributions indépendantes, on va pouvoir combiner (3.6) et (3.8) pour obtenir :

$$(\mathsf{SK}_F \| \mathsf{SK}_W, \mathsf{HD}) \approx_{s,\epsilon} (U_l, \mathsf{HD}).$$

À partir de sk_F et sk_W les secrets *fatal* et *warning*, le secret FAST consiste en $\mathsf{sk} = \mathsf{H}_l(\mathsf{sk}_F \| \mathsf{sk}_W)$. Par la sécurité de l'oracle aléatoire H_l ainsi appliqué à $\mathsf{SK} \stackrel{\text{def}}{=} \mathsf{SK}_F \| \mathsf{SK}_W$, il en résulte que :

$$(\mathsf{SK}, \mathsf{HD}) \approx_{s,\epsilon} (U_l, \mathsf{HD}).$$

□

Et si les attributs sont corrélés ? L'hypothèse **Hyp2** stipule que les valeurs les valeurs d'attributs du profil Ω sont indépendantes et donc non corrélées. Si on considère désormais que les variables $WA_1, \ldots, WA_n$ sont corrélées, nous discutons selon les cas :

- pour les secrets sont statiques, on obtient une approximation similaire à 3.4 par la sécurité de l'oracle aléatoire ;
- lorsque les secrets sont variables, on retombe sur le cas où $\rho \leqslant n$ secrets corrélés vont potentiellement être appelés par le même FE. En supposant l'usage de FEs réutilisables sur lesdits attributs, on obtiendrait alors une approximation analogue à celle de la Définition 42. Or, pour obtenir la sécurité du module FAST, il nous faudrait plutôt une approximation du type :
  $(\mathsf{sp}^1, \ldots, \mathsf{sp}^\rho, P^1, \ldots, P^\rho) \approx_{s,\epsilon} (\mu^1, \ldots, \mu^\rho, P^1, \ldots, P^\rho)$, où les μ_is sont des éléments aléatoires.

Pour l'instant, nous conservons l'hypothèse **Hyp2** mais nous verrons au cours du Chapitre 4 comment s'en affranchir afin d'assurer la sécurité du module FAST même dans le cas où les attributs considérés sont corrélés.

3.5 Analyse de la réutilisabilité

Dans cette section, nous allons montrer que la sécurité des outils mis en jeu dans l'instanciation définie par les Figures 3.6, 3.7 et 3.8 satisfait à la réutilisabilité (Définition 50 et Figure 3.2).

Soit $\mathcal{A}$ un adversaire calculatoire capable de procéder à $q = \mathsf{poly}(\lambda)$ appels à l'oracle l'oracle $\mathcal{O}^{\text{corrupt}}$ avec $q \leqslant \rho$.

Note. Si $(\mathsf{Gen}, \mathsf{Rep})$ est FE (ρ, ϵ, s)-réutilisable alors, en reprenant la Définition 42 et la notation associée, on a : $\delta^{\mathcal{D}_s}((R^1, \ldots R^{j-1}, R^j, R^{j+1}, \ldots, R^\rho, P^1, \ldots, P^\rho), (R^1, \ldots R^{j-1}, \mu, R^{j+1}, \ldots, R^\rho, P^1, \ldots, P^\rho)) \leqslant \epsilon$. En particulier, pour tout $q \leqslant \rho$, on aura aussi : $\delta^{\mathcal{D}_s}((R^1, \ldots R^{j-1}, R^j, R^{j+1}, \ldots, R^q, P^1, \ldots, P^\rho), (R^1, \ldots R^{j-1}, \mu, R^{j+1}, \ldots, R^q, P^1, \ldots, P^\rho)) \leqslant \epsilon$.

Théorème 2. *Les Figures 3.6, 3.7 et 3.8 définissent un module FAST q-réutilisable satisfaisant la Définition 50.*

Proof. Partant du jeu défini par la Figure 3.2 (b) et dénoté ici $\mathcal{G}_0$, nous proposons un enchaînement de jeux indistinguable, du point de vue de l'adversaire $\mathcal{A}$, qui nous permettra de conclure directement sur la sécurité de notre instanciation FAST.
Avant d'entrer dans les détails et en reprenant les notations déjà définies, tout profil Ω^j s'écrira $(\mathsf{att}_1^j, \ldots, \mathsf{att}_n^j)$ où pour tout $i = 1 \ldots n$, $\mathsf{att}_i^j = (\mathsf{id}_i, w_i^j, b_i)$.

Remarquons alors que, pour j fixé, les w_i^js sont les valeurs des différents attributs du profil Ω^j ; les exposants réfèrent aux différents profils. Au contraire, pour i fixé, les w_i^js

constituent des valeurs issues du même identifiant d'attribut pour un même utilisateur et seront donc potentiellement corrélées ;

$\mathcal{G}_0$ Ce jeu consiste en $\mathrm{Dec}^1_{\mathcal{C},\mathcal{A}}(\lambda)$. Le challenger $\mathcal{C}$ et l'adversaire $\mathcal{A}$ interagissent donc comme prescrit par la Figure 3.2 (b) (cas où $b = 1$). Après avoir décidé d'attaquer l'exposant j_0, $\mathcal{C}$ retournera $r = \mathsf{sk}^{j_0}$ à $\mathcal{A}$ puisque $b = 1$. Ainsi, $\mathcal{A}$ connaît L, SC et r à la fin du jeu . Sans perte de généralité et à des fins rédactionnelles, on supposera que les q requêtes de $\mathcal{A}$ à l'oracle ont été appliquées aux exposants $1, 2, \ldots q$. Ainsi, $\mathcal{A}$ a finalement obtenu :

$$\mathsf{sk}^1, \ldots, \mathsf{sk}^q, r = \mathsf{sk}^{j_0}, \mathsf{hd}^1, \ldots, \mathsf{hd}^\rho. \tag{3.9}$$

où sk^{j_0} s'écrit $\mathsf{sk}^{j_0} = \mathsf{H}_l(\mathsf{sk}_F^{j_0} \| \mathsf{sk}_W^{j_0})$ (voir Figure 3.7, étapes 2.a et 3.a pour plus de détails).

$\mathcal{G}_1$ Ce jeu se déroule comme le précédent sauf en ce qui concerne le traitement de l'exposant j_0 par $\mathcal{C}$. Au lieu d'exécuter honnêtement GenKey sur Ω^{j_0} caractérisé par le vecteur de valeurs $(w_1^{j_0}, \ldots, w_n^{j_0})$, $\mathcal{C}$ en modifie l'étape 2 dans le cas des attributs estampillés *fatal*. Pour tout $i \in \mathsf{LF}$, $\mathcal{C}$ commence par calculer honnêtement $(\mathsf{sp}_i^{j_0}, \mathsf{hd}_i^{j_0}) \leftarrow \mathsf{DC}[i](w_i^{j_0})$ (Figure 3.7, étape 2.a). Au lieu d'utiliser honnêtement les $\mathsf{sp}_i^{j_0}$, $\mathcal{C}$ choisit aléatoirement $\mu_i \xleftarrow{\$} \{0,1\}^{l_i}$ et forme $\bar{\mathsf{sk}}_F^{j_0} = \mu_{i_1} \| \ldots \| \mu_{i_f}$. Avant de poursuivre le déroulement du jeu, regardons l'ensemble des profils de manière transverse en considérant les données retournées par $\mathsf{DC}[i]$ appliqués à $w_i^1, \ldots, w_i^{j_0}, \ldots, w_i^\rho$. Lorsque att_i est un attribut statique (*resp.* variable), la sécurité de H_i (*resp.* la réutilisabilité de $(\mathsf{Gen}_i, \mathsf{Rep}_i)$) assure que $(\mathsf{sp}_i^1, \ldots, \mathsf{sp}_i^\rho, \mathsf{hd}_i^1, \ldots, \mathsf{hd}_i^\rho)$ est indistinguable de $(\mathsf{sp}_i^1, \ldots, \mathsf{sp}_i^{j_0-1}, \mu_i, \mathsf{sp}_i^{j_0+1}, \ldots, \mathsf{sp}_i^\rho, \mathsf{hd}_i^1, \ldots, \mathsf{hd}_i^\rho)$.

Reprenant le jeu, $\mathcal{C}$ forme $\bar{\mathsf{sk}}_F^{j_0} \| \mathsf{sk}_W^{j_0}$, où $\mathsf{sk}_W^{j_0}$ est généré honnêtement (Figure 3.7, étape 3.a.) de sorte que l'adversaire $\mathcal{A}$ aura finalement obtenu :

$$\mathsf{sk}^1, \ldots, \mathsf{sk}^q, r = \bar{\mathsf{sk}}^{j_0}, \mathsf{hd}^1, \ldots, \mathsf{hd}^\rho. \tag{3.10}$$

où $\bar{\mathsf{sk}}^{j_0} = \mathsf{H}_l(\bar{\mathsf{sk}}_F^{j_0} \| \mathsf{sk}_W^{j_0})$.

L'indistinguabilité rappelée ci-dessus et implique que 3.9 et 3.10 sont indistinguables. Ceci assure l'indistinguabilité des jeux $\mathcal{G}_1$ et $\mathcal{G}_0$ aux yeux de $\mathcal{A}$.

$\mathcal{G}_2$ Ce jeu se déroule comme le précédent jusque dans le traitement des attributs de type *fatal* du profil Ω^{j_0}. Puis, le challenger $\mathcal{C}$ modifie également le traitement de attributs estampillés *warning*. Pour tout $i \in \mathsf{LW}$, il calcule honnêtement $(\mathsf{sp}_i^{j_0}, \mathsf{hd}_i^{j_0})$ que l'attribut att_i soit statique ou variable pour

former $w_W^{j_0} = \{w_{j_1}^{j_0}, \ldots, w_{j_w}^{j_0}\}$. Puis, $\mathcal{C}$ calcule honnêtement $(\mathsf{sk}_W^{j_0}, \mathsf{hd}_{n+1}^{j_0} = \mathsf{Gen}(w_W^{j_0}, t_{n+1})$ et tire aléatoirement $\mu_W \leftarrow \{0,1\}^{l_{n+1}}$. La réutilisabilité de $\mathsf{Gen}_{n+1}, \mathsf{Rep}_{n+1})$ assure que $(\mathsf{sk}_W^1, \ldots, \mathsf{sk}_W^\rho, \mathsf{hd}_{n+1}^1, \ldots, \mathsf{hd}_{n+1}^\rho)$ est indistinguable de $(\mathsf{sk}_W^1, \ldots, \mathsf{sk}_W^{j_0-1}, \mu_W, \mathsf{sk}_W^{j_0+1}, \ldots, \mathsf{sk}_W^\rho, \mathsf{hd}_{n+1}^1, \ldots, \mathsf{hd}_{n+1}^\rho)$. $\mathcal{C}$ peut alors former $\bar{\mathsf{sk}}_F^{j_0} \| \mu_W$ où $\bar{\mathsf{sk}}_F^{j_0} = \mu_{i_1} \| \ldots \| \mu_{i_f}$ et μ_W sont des valeurs aléatoires par les déroulements respectifs des jeux $\mathcal{G}_1$ et $\mathcal{G}_2$.

Ainsi $\mathcal{A}$ a finalement obtenu :

$$\mathsf{sk}^1, \ldots, \mathsf{sk}^q, r = \bar{\mathsf{sk}}^{j_0}, \mathsf{hd}^1, \ldots, \mathsf{hd}^\rho. \tag{3.11}$$

où $\bar{\mathsf{sk}}^{j_0} = \mathsf{H}_l(\bar{\mathsf{sk}}_F^{j_0} \| \mu_W)$.

L'indistinguabilité rappelée ci-dessus implique que 3.10 et 3.11 sont indistinguables. Ceci assure l'indistinguabilité des jeux $\mathcal{G}_3$ et $\mathcal{G}_2$ aux yeux de $\mathcal{A}$.

$\mathcal{G}_3$ Ce jeu reprend le précédent et est principalement introduit à des fins de clarté. La méthodologie de $\mathcal{G}_2$ amène $\mathcal{C}$ à renvoyer $r = \mathsf{H}_l(\mu_{i_1} \| \ldots \| \mu_{i_f} \| \mu_W)$. Au cours du jeu courant, $\mathcal{C}$ choisit $\mu \xleftarrow{\$} \{0,1\}^l$ et fixe $r = \mu$. La sécurité de l'oracle aléatoire H_l assure que $\mathsf{H}_l(\mu_{i_1} \| \ldots \| \mu_{i_f} \| \mu_W)$ et μ sont indistinguables.

Ainsi, $\mathcal{A}$ aura finalement obtenu :

$$\mathsf{sk}^1, \ldots, \mathsf{sk}^q, r = \mu, \mathsf{hd}^1, \ldots, \mathsf{hd}^\rho. \tag{3.12}$$

où $\mu \xleftarrow{\$} \{0,1\}^l$. L'indistinguabilité rappelée ci-dessus permet de conclure que 3.11 et 3.12 sont indistinguables. On en conclut que $\mathcal{G}_3$ et $\mathcal{G}_2$ sont indistinguables aux yeux de l'adversaire $\mathcal{A}$. Remarquons par ailleurs, que $\mathcal{G}_3$ consiste en $\mathrm{Dec}_{\mathcal{C},\mathcal{A}}^0(\lambda)$.

Par une succession de jeux, nous avons montré que $\mathrm{Dec}_{\mathcal{C},\mathcal{A}}^0(\lambda)$ et $\mathrm{Dec}_{\mathcal{C},\mathcal{A}}^1(\lambda)$ sont indistinguables aux yeux de l'adversaire $\mathcal{A}$. Cette indistinguabilité implique trivialement la difficulté pour $\mathcal{A}$ de remporter la version calculatoire du jeu Figure 3.2 (a).

Note. À partir du jeu $\mathcal{G}_1$, le challenger doit supposer que $\mathcal{A}$ va effectivement attaquer l'exposant j_0 pour pouvoir en modifier le traitement : cela arrive avec probabilité non négligeable $(1/\rho)$. Il suffit de recommencer le jeu si tel n'est pas le cas.

□

3.6 Exemple de profil au stade TA2

Cette section présente un exemple de profil utilisateur que le stade TA2 de la solution pourra exploiter en répondant aux limitations du stade TA1 (figure 2.9 respectant les

requis précédemment rappelés à la figure 2.9. Afin de répondre aux limitations L0 et L1, nous avons considéré le domaine du *device fingerprinting* dont la pertinence est soulignée par la section 3.1 alors que le module FAST exhibe une manière d'en tirer parti afin de répondre aux limitations L2 et L3. Dans cette section, nous présentons un exemple de profil utilisateur, qui couplé avec FAST, permettra l'accession au stade TA2. Afin d'assurer la sécurité de notre solution, ce profil devra porter suffisamment d'entropie. L'étude proposée dans [111] traite le sujet de l'identification des utilisateurs sous un autre angle, il n'est pas proposé d'estimation de l'entropie porté par chacune des empreintes d'appareil étudiées.

Les résultats présentés ici sont le fruit d'études internes réalisées sur quelques dizaines d'utilisateurs et devront être menées à grande échelle pour être considérées comme fiables et permettre une industrialisation. Nous avons alors estimé, de manière grossière, l'entropie portée par chacun des attributs constituant un profil au stade TA2 . Pour notre défense, cette estimation sera une minoration de l'entropie réelle. Ce point nécessitera des études futures pour que l'accession au stade TA2 soit complète.

3.6.1 De nouveaux attributs

En 2014, nous avons considéré certains attributs (listes des contacts, des applications) comme de potentiels attributs, idée reprise et confortée par les études présentées à la section 3.1.

Ainsi dans l'accession au stade TA2, nous proposons d'intégrer à la constitution d'un profil utilisateur (Figure 2.2), les attributs suivants :

- **Top15_contacts.** La liste des 15 contacts les plus fréquents de l'utilisateur $\mathcal{U}$ (messages et appels confondus). À titre d'exemple, le prénom le plus donné aux États-Unis est James [148] avec une fréquence de 1.518% menant à min-entropie d'environ $-log(1.518.10^{-2}) \approx 6$ bits. En supposant communs certains contacts familaux ("Dad", "Mom", ...), nous considérons alors que seulement 10 contacts parmi la liste des 15 apportent de l'information. Soit une entropie de $6 \times 10 = 60$ bits.

- **Top_apps.** La liste des applications de l'utilisateur. Nos études internes nous ont poussé à omettre certaines applications trop communes–parce qu'inhérentes au sytème ou trop fréquentes pour être considérées comme différenciantes (gmail, gmaps, facebook, ...)–, nous éliminons alors les 10 applications les plus utilisées dans nos considérations. Ainsi, l'information minimale apportée par chacune des autres applications est estimée à 4 bits. Si un utilisateur a trop d'applications (¿50), on définit alors Top30_apps qui, en reprenant la discussion sur les 10 fréquentes mises de côté, forme un attribut d'environ $4 \times 20 = 80$ bits;

- **Top20_chansons.** La liste des 20 chansons les plus jouées en local sur $\mathcal{T}$, ou via une application musicale. De manière analogue à Top_apps, on fixe à 4 bits la min-entropie portée par le titre d'une chanson pour former attribut portant alors 120 bits d'entropie. On pourrait, de manière analogue à Top_apps, définir Top_chansons, où la taille de l'attribut ne serait pas fixé.

Remarque 2. *En ce qui concerne les attributs Top_apps et Top20_chansons, remarquons que :*

- *si les préférences de l'utilisateur en matière de musique ou de jeux sont connus, cela constituera un avantage pour deviner la valeur de l'attribut. Cela peut se voir comme l'information auxiliaire I dans le cas de FEs dans le cas moyen ;*
- *de manière analogue, si un élément de la liste venait à être révélé, il pourrait alors trahir les préférences de l'utilisateur.*

Pour ces attributs, comme pour tous les autres tout au long de ce chapitre, les valeurs sont considérées comme étant des secrets de l'utilisateur.

Au cours de nos travaux, nous avons également exploité deux autres attributs, notés respectivement AttBin et Hotspots que nous décrivons ci-dessous.

Attribut AttBin

Cet attribut permet de revoir le traitement des attributs ne portant qu'un seul bit d'information. En effet, certains des attributs récoltés au stade TA1 ont pour valeur "oui" ou "non". Une liste non exhaustive pourrait être la suivante : "Mode root", "WiFi activé", "Mode Déboggage" (Debug), "Détection Émulateur" (DE), "Mode Développeur" (Dev), Écran Allumé" (EA), "Vérouillage Téléphone" (VT), "Écran Allumé en Charge" (EAC)....

Numérotant ces attributs, nous définissons l'attribut AttBin par la Figure 3.9.

Attribut	Root	WiFi	Debug	DE	EA	VT	Dev	...	EAC
Valeur	non	oui	oui	non	oui	non	non	...	non
AttBin	0	1	1	0	1	0	0	...	0

Figure 3.9: Attribut AttBin

La valeur prise par AttBin consiste alors en une chaîne binaire de longueur n pour laquelle le i$^{\text{ième}}$ bit est un "1" si l'attribut numéroté i a pour valeur "oui" et un "0" si la valeur est "non". En supposant que ces différents détecteurs binaires sont indépendants, nous estimons la min-entropie portée par l'attribut AttBin à n bits. En l'état, nous avons identifié une trentaine de détecteurs binaires, redondants pour beaucoup et donc

portant seulement une dizaine de bits d'entropie. À terme, l'idée serait d'en obtenir plus pour obtenir une entropie plus importante. Un tel attribut sera alors traité par un FE réutilisable et basé sur la distance de Hamming. Rappelons, si nécessaire, que le seul candidat actuel est la récente construction de Canetti *et al.* [94].

Attribut Hotspots

Étudié dans [111], la liste des bornes Wi-Fi environnantes nous apparaît comme un attribut à fort potentiel. L'idée est de considérer la liste des n' bornes Wi-Fi avec le plus fort signal aux environs du smartphone utilisateur. La valeur d'attribut consiste alors en un ensemble de SSIDs (*Service Set Identifier*), noté ssid. Un SSID est le nom d'un réseau Wi-Fi, encodé sur 32 octets [149] offrant potentiellement $8^{32} = 2^{96}$ possibilités de nommage pour un seul SSID. Un bon nombre d'utilisateurs, si ce n'est la majorité, ne renomme pas son SSID allant ainsi à l'encontre des bonnes pratiques [149, 150] ; d'autre part, tout renommage ne saurait être réellement aléatoire. Il ne faudra donc pas considérer que chaque SSID porte 96 bits d'entropie. À part en connaissant la location de l'utilisateur, ce qui irait à l'encontre de la remarque 2 et constituerait la réalisation d'une attaque ou *exploit*, les noms des différents SSIDs constituant ssid sont indépendants. Il est donc légitime de considérer que ssid constituera un attribut à très forte entropie.

Remarquons au passage qu'un tel attribut permet d'associer plusieurs profils utilisateur en fonction de sa localisation (Maison, Travail, Vacances, ...). Nous fixons à 10 bits la min-entropie portée par un SSID. Ainsi, en fixant à 10 le nombre de SSIDs sélectionnés, l'attribut ssid portera 100 bits d'entropie.

3.6.2 Exemple de profil et temps de calcul

En ajoutant les attributs présentés ci-dessus à un profil tel que défini par le stade TA1, nous allons constituer un profil portant plus d'entropie.

Autres attributs et entropie

Nous commençons par estimer l'entropie portée par les attributs du stade TA1, parfois de manière cavalière car nous n'avons pu mener d'études à grande échelle alors que certaines données ne sont publiquement disponibles.

- **Application App.** Nous faisons la différence entre les différentes versions de l'application App. Notre version la plus fréquente apparaît chez environ 7% des utilisateurs finaux. Cela mène à environ $-log(0.07) \approx 4$ bits ;
- **OpenSSL.** Une étude similaire sur les version embarquée de OpenSSL mène à estimer la min-entropie à environ 4 bits ;

- **IMSI.** Les 8 derniers digits de l'IMSI, stocké dans la carte SIM de l'appareil considéré, sont aléatoires et permettent d'identifier uniquement un appareil [151, 152] : cet attribut porte donc $8 \times log(10) \approx 26$ bits d'information ;
- **Nom du téléphone.** Dans ce cas, les résultats remontés sont très variables. Bien souvent, les utilisateurs ne renomment pas leur téléphone qui aura alors pour nom le modèle considéré. Nous estimons à 5 bits cette information.

Nous proposons alors un exemple de profil utilisateur au stade TA2 à travers la Figure 3.10. Bien que présents car historiques, les attributs marqués d'un $-$ sont des attributs pour lesquels nous considérons que l'entropie portée est négligeable devant celle des autres. Un tel profil mène à une entropie globale d'environ 400 bits.

Attribut att_i	Exemple Valeur ω_i	Min-Entropie (bits)	Fat. (F)/Warn. (W)	t_i
IMEI	354784054129987/02	47	F	0
IMSI (SIM)	208 20 11 34567890	26	F	0
Connecteur USB	micro USB	$-$	$-$	$-$
Définition écran	720×1280	$-$	$-$	$-$
Debogueur	ADB	$-$	$-$	$-$
Hash OpenSSL	8e52d5cb...20afb443	4	W	0
Nom du téléphone	Téléphone de Bob	5	W	0
Hash de App	0d04bfeb...34c9984d	5	F	0
Top15_contacts	$\{\text{Alice}, \text{Bob}, \ldots \text{Eve}\}$	60	F	4
Top30_Apps	$\{\text{App}_1, \ldots, \text{App}_{15}\}$	80	W	8
Top20_Chansons	$\{\text{Ch}_1, \ldots, \text{Ch}_{20}\}$	80	W	8
AttBin	$v \in \{0,1\}^{30}$	10	W	3
Hotpsots	$\{\text{ssid}_1, \ldots, \text{ssid}_{10}\}$	100	F	3

Figure 3.10: Exemple de Profil Utilisateur au stade TA2

Temps de calcul

Puisque les estimations d'entropie ne sont pas encore formalisées, nos implémentations se limitent à un *Proof of Concept* (PoC), implémenté en C. D'autre part, les travaux présentés au cours de ce chapitre ont eu lieu en parallèle avec ceux du chapitre 4 où nous résoudrons la réutilisabilité des FEs basés sur la différence d'ensemble proposés Dodis *et al.* [13]. Ce sont ces constructions plus efficaces – en stockage et en temps d'exécution (voir chapitre 4)– que celles proposées par Canetti *et al.* [94] que nous avons alors implémentées. Lesdits FEs sont basés sur des secure sketchs couplés a des extracteurs forts. Suivant les travaux de Krawczyk [37], l'extracteur calculatoire sera joué par la construction HMAC-SHA256 dont la sortie sera tronquée à la longueur spécifiée par les paramètres l et l_is (Figures 3.6, 3.7 et 3.8).

Spécifications (Chapitre 2) et utilisateurs acceptent qu'un enrôlement ne osit pas instantané ; en revanche, le processus global d'une session d'authentification ne devra pas dépasser les 400 ms. Note solution actuelle alloue donc une fenêtre de temps de l'ordre de 100 ms à FAST pour re-générer la clé sk via RepKey.

Précisions et notation Les travaux de Dodis *et al.* proposent deux types de FEs basés sur la différence d'ensembles :

- Le cas où la taille de l'attribut n'est pas fixée (Top_apps) est géré par la Construction 6 des travaux de Dodis *et al.* [13]. Dans ce cas, insertion et/ou délétion constitue une erreur lors qu'une substitution se verra comme deux erreurs (insertion + délétion) ;
- Lorsque la taille de l'attribut est fixée en amont (Top15_contacts, Top20_chansons), le FE considéré est la Construction 5 des mêmes travaux [13], qui consiste en une amélioration du schéma proposé par Juels et Sudan [15]. Dans ce cas, une substitution consiste en une erreur.

Pour le profil dépeint à la figure 3.10, on obtient les temps de calcul résumés à la Figure 3.11.

Attribut/Profil	Outil	Paramètres du FE $(\mathcal{M}, m, l, t)$	Temps (ms)
Top15_contacts	FE [13], Sec. 6.2	$(\mathsf{SDif}(\mathbb{F}_{2^{12}}), 60, 40, 4)$	40
Top_apps (Top30_apps)	FE [13], Sec. 6.3, (6.2)	$(\mathsf{SDif}(\mathbb{F}_{2^{12}}), 80, 40, 8)$	50
Top20_chansons	FE [13], Sec. 6.2	$(\mathsf{SDif}(\mathbb{F}_{2^{12}}), 80, 40, 8)$	60
Hotspots	FE [13], Sec. 6.2	$(\mathsf{SDif}(\mathbb{F}_{2^{12}}), 100, 60, 3)$	40
AttBin	FE [94], Sec. 4	non fait	–
Profil Ω (Figure 3.10)	FAST (Figures 3.6, 3.7 et 3.8)	$\mathsf{SDif}_4(\mathbb{F}_{2^{20}})$, $l_{n+1} = 256$, $t_{n+1} = 2$	220

Figure 3.11: Outils et temps de calcul sur un Samsung Galaxy S6.

Remarque 3. *Les attributs estampillés warning sont le nom du téléphone, la version de OpenSSL, Top_apps et Top20_chansons. Pour les deux derniers, les secrets partiels sont longs de 40 bits. Nous les découpons pour en former 4 de longueur 20 correspondant alors aux paramètres de* $(\mathsf{Gen}_{n+1}, \mathsf{Rep}_{n+1})$. *L'attribut warning* w_W *se voit alors comme un ensemble de 6 éléments et on adapte la tolérance qui passe alors à 3.*

Cette méthodologie permet de rester sur des corps de cardinal raisonnable (2^{20} *contre* 2^{40}*) et limite donc les temps de calcul.*

Précisons que les temps de calculs induits par les attributs statiques et gérés par HMAC-SHA256 sont négligeables devant le temps de reproduction de clés par les FEs. Nous ne les avons pas détaillé ici mais ils sont pris en compte dans la ligne relative à Ω. Nous concluons lors que les temps d'exécution sont trop importants au vu de la contrainte de 100 ms rappelée ci-dessus.

3.7 Limitations au stade TA2

À la fin de ce chapitre, théorie et pratique mènent à cinq limitations :

- **Pratiques.** La section précédente a mise en exergue que nous méconnaissons les attributs manipulés et l'entropie qu'ils portent. Ce faisant, nous pourrons affiner les paramètres choisis pour finalement optimiser nos implémentations et atteindre notre contrainte de temps de l'ordre de 100 ms alors que notre PoC propose des temps de reconstruction de clés de l'ordre de 200 ms sur un Samsung Galaxy S6.
- **Théoriques.** En plus d'être un problème ouvert depuis des années (Boyen 2004), les Sections 3.5 et 3.6souligne l'intérêt dans nos travaux de manipuler des FEs réutilisables basés sur la différence d'ensembles dans le cas du *grand univers*. D'autre part, les études menées en interne ont vite exhibé que la durée de vie des clés issues par FAST étaient relativement réduite. Ceci est dû au fait que certains attributs (Top_apps, Top20_chansons,...) varient continuellement dans le temps au lieu de rester proches d'une valeur référence comme c'est le cas de la biométrie (physiologique).
- La cinquième limitation est à la fois pratique et théorique. Notre preuve de sécurité pour FAST requiert des attributs indépendants. Or, ce n'est pas le cas dans les faits : à titre d'exemple, les préférences musicales d'un utilisateur pourront influer sur les applications musicales qu'il aura installées.

Nous explicitons désormais ces limitations avant de les résumer à la figure 3.12.

3.7.1 Limitations L4 et L5 ou un manque de pratique

Même si de nombreuses études ont été proposées ces dernières années, le domaine du *fingerprinting* et tout particulièrement le domaine du *fingerprinting* d'appareil est un domaine très récent pour lequel l'entropie contenue dans les attributs que nous considérons n'est pas encore exactement connue. Avant de passer à une phase d'industrialisation, notre plan d'action est d'installer des versions bêta chez des utilisateurs de l'application App dans sa version décrite au stade TA1, qui permettra aussi de récolter ces attributs variables nouvellement considérés. Anonymiser et remonter ces valeurs sur les serveurs Worldline permettra alors de procéder à des études statistiques.

D'autre part, le *fingerprinting* et le respect de la vie sont des mots-clés de plus en plus utilisés dans le monde industriel, il n'est pas déraisonnable de penser que des études complémentaires aient lieu dans un futur proche. Nous notons cette méconnaissance des attributs considérés et de l'entropie précise qu'ils contiennent la limitation L4.

Même si, nous l'avons vu, nos première estimations sont grossières, nous sommes alors confrontés à des premières contraintes de temps. Alors que sur un PC classique, les calculs prennent moins de 100 ms, le passage sur smartphone –Galaxy S6 de surcroît– entraîne des temps de calcul multipliés par 2 (Figure 3.11). Nous notons L5 cette limitation.

3.7.2 Limitation L6 ou la gestion de la différence d'ensembles

À la lecture de ce chapitre, il apparaît clairement que la métrique qui nous est la plus pertinente la différence d'ensembles. En effet, les empreintes les plus exploitées sont toutes sous forme de d'ensembles (applications, plugins, contacts, chansons, cookies, polices utilisées, ...). D'autre part, à l'exception de l'iris [153], la distance de Hamming historiquement considérée n'est pas adaptée aux empreintes biométriques [154, 155]. Puisque corrigeant des séquences de caractères, elle dépend de leur ordre et la suppression d'un seul caractère entre deux chaînes pourra conduire à d'importantes différences. Or, il n'est pas rare que les systèmes biométriques présentent des effacements et des valeurs sous forme d'ensemble de caractéristiques pour lesquelles l'ordre ne doit pas être pris en compte [156, 154, 13, 15].

Comme nous l'avons vu, la seule possibilité pour obtenir un FE réutilisable et basé sur la distance la différence d'ensembles consiste à convertir les travaux de Canetti *et al.* [94] via la l'équivalence bin-set. Malheureusement, cette équivalence ne s'applique que dans le cas *petit univers* où les éléments constituant la valeur w proviennent d'un univers de cardinal polynomial en la longueur de w (*i.e.* l'univers est de cardinal $\mathsf{poly}(|w|)$). Ce cas n'est pas naturel puisque la plupart des attributs considérés sont liés à des univers de cardinal potentiellement superpolynomial en $|w|$ (chansons, applications, ...) ; de plus ce scénario impose de connaître la taille de l'univers en amont et à l'énumérer pour en permettre la conversion binaire. Finalement, le *large universe setting* pour lequel il n'existe aucun FE réutilisable apparaît pourtant adapté aux exemples sus-cités et aux considérations de la littérature [15, 13, 91, 92] rappelées au chapitre 1 (sous-section 1.5.2).

La conception d'un FE basé sur la différence d'ensemble dans le contexte du *grand univers* est aujourd'hui un problème ouvert, limitant ainsi l'exploitation du module FAST. Nous notons L6 cette limitation.

3.7.3 Limitation L7 ou la durée de vie réduite des attributs

Nous avons vu que le *fingerprinting* constitue une première réponse aux limitations L0 et L1 (Figure 2.9). En plus d'exhiber le besoin de concevoir un FE réutilisable dans le contexte du *grand univers*, nos premiers tests ont vite mis en lumière une autre limitation. Bien qu'identifiants, ces attributs souffrent d'une durée de vie raccourcie en comparaison à la biométrie classique. En effet, même si les empreintes biométriques

classiques présentent des variations inhérentes, leur valeur globale reste semblable au cours du temps contrairement aux attributs considérés dans le cas du *fingerprinting*. Par exemple, une liste de contacts peut fortement varier au cours d'une longue période tout en assurant l'authenticité d'un utilisateur si ces changements sont continuellement pris en compte. Par conséquent, les clés extraites par le module FAST pourront souffrir d'une durée de vie raccourcie. Puisque des sessions de ré-enrôlements sont toujours critiques (considérations sécuritaires, charges serveurs, expérience utilisateur appauvrie), il faudra être capable de minimiser leurs occurrences. Cette limitation sera notée L7.

3.7.4 Limitation L8 ou non-indépendances des attributs

Cette limitation est liée à la limitations L4. Notre méconnaissance des attributs ne nous permettra pas de conclure que deux attributs donnés sont indépendants. Bien évidemment, nous faisons fi ici des attributs pour lesquels la question ne se pose pas (IMEI, IMSI, nom du téléphone, ...). Et si jamais deux attributs sont corrélés, nous devrions tout de même pouvoir les exploiter puisqu'ils seront malgré tout différenciant et porteurs d'entropie. Cette limitation sera notée L8.

La Figure 3.12 constitue un récapitulatif des limitations induites par le stade TA-2.

Limitation	Définition
L4	Méconnaissance des attributs nouvellement identifiés.
L5	Optimisation des temps de calculs.
L6	Conception d'un FE réutilisable basé sur la différence d'ensembles dans le contexte du *grand univers*.
L7	Gestion des attributs qui varient au cours du temps.
L8	Gestion de la non-indépendance des attributs.

Figure 3.12: Limitations du stade TA2

3.8 Conclusion

Reposant sur des FEs, nous avons proposé une instanciation du module FAST permettant de répondre aux limitations L0, L1, L2 et L3 tout en continuant de satisfaire les requis édités par la solution TA (Figure 2.8) permettant l'accession au stade TA2. Cependant, de nouvelles limitations ont été identifiées et devront être résolues pour espérer une prochaine industrialisation. Une stratégie concernant les limitations pratiques (L4 et L5) sera de proposer une version bêta de l'application App fonctionnant selon le stade TA1 tout en remontant les valeurs des attributs nouvellement considérés pour permettre de réaliser des statistiques à grande échelle.

Nous nous intéressons aux limitations théoriques L6, L7 et L8 au cours du prochain Chapitre qui, en particulier, nous verra résoudre la réutilisabilité des FEs dans le cas de la différence d'ensembles. Nous y proposerons également framework générique permettant d'assurer la réutilisabilité de tout FE et le concept d'*Adaptive Fuzzy Extractor* pour répondre à la limitation L7. Nous verrons également que notre méthode de résolution pour L6 permettra également de répondre à L8. Ces résolutions permettrons l'accession au stade TA3.

Chapitre 4 :

Stade TA3 - FEs réutilisables et adaptatifs. Instanciation pour la différence d'ensembles.

Contents

Comme rappelé au Chapitre 1, bien que prometteurs les FEs souffrent de limitations notamment au de sujet leur réutilisabilité et des données auxquelles la communauté a cherché à les appliquer. En ce qui concerne le premier point, Canetti *et al.* ont récemment proposé le premier FE réutilisable ; leur construction est basée sur la distance de Hamming et voit sa sécurité reposer sur le modèle de l'oracle aléatoire. D'autre part, nous avons vu au Chapitre 3 que les PUFs et la biométrie qui sont les deux cas d'usage les plus avancés dans le cas des FEs ne bénéficient pas d'une instanciation sans heurt. Conséquemment à des études internes et aux résultats encourageants de la communauté [110, 111], nous avons alors proposé le domaine du *fingerprinting* comme un nouveau cas d'usage. Cependant, bon nombre de ces empreintes d'appareils et/ou de navigateur apparaissent sous la forme de listes alors que la conception d'un Fuzzy Extractor réutilisable basé sur la différence d'ensembles [1] restait alors un problème ouvert.

[1] La plupart des cas concrets basés sur la différence d'ensemble sont naturellement liés au cas du *grand univers* et ne peuvent donc pas être gérés par l'équivalence bin-set.

Le but de ce chapitre est donc de répondre à cette problématique : pour ce faire, nous proposons une méthodologie générique qui permet d'obtenir un FE réutilisable à partir de tout FE classique. Pour ce faire, nous introduisons le concept d'isométries pseudo-entropique réutilisable ou *Reusable Pseudoentropic Isometry* qui projettent tout élément d'un espace métrique vers un autre espace métrique tout en préservant distance et entropie. Ainsi, combinant ce concept avec tout FE classique nous permettra de concevoir un FE réutilisable.

Nous proposons alors une instanciation prouvée réutilisable dans le cas de la différence d'ensembles[2], résolvant un problème ouvert depuis 2004 et les travaux de Boyen [85]. Ce résultat permettrait alors d'instancier notre module FAST proposé au Chapitre précédent de manière sécurisée.

La suite de ce chapitre, écrite en anglais, décrit ces travaux qui sont actuellement en soumission et en accès public sur ePrint [5]. Cette publication est le résultat d'un travail conjoint avec Paul-Edmond Berthier, Chloé Cachet, Stéphane Cauchie, Benjamin Fuller et Philippe Gaborit.

Our contribution

The contribution of our work is threefold:

1. We propose a generic framework to address reusability that can be instantiated with any nonreusable FE. The idea is to preprocess fuzzy secrets while maintaining distances between two noisy readings. More precisely, when ρ enrollments have to be done from the fuzzy secret w, such a process will output ρ unrelated values $\Omega^1, \ldots, \Omega^\rho$ that can then be securely used once by any nonreusable FE. Now, if a user wants to authenticate herself toward provider j from a noisy reading w', our pre-processing function will generate Ω'_j such as the distance between Ω'_j and Ω^j is the same as between w and w'. We capture this preprocessing by a new primitive we call *reusable pseudoentropic isometric* (RPI). Informally, a RPI pseudorandomly projects fuzzy secrets while maintaining distances between two noisy readings.

2. Relying on the previously introduced framework, we construct a RPI for the set difference metric which leads us to design the first practical set difference-based reusable FE in the large universe setting. Furthermore, we will see that our construction will enjoy efficient time and storage complexities of the best nonreusable FEs while enabling an error correction linear in n.

[2]Nous nous concentrerons sur le cas du *grand univers* dont la pertinence a été discutée au cours du Chapitre 1.

3. Considering fingerprints with deeper variations (*e.g.* behavioral, device and browser fingerprints) but still identifying [157, 110, 133, 111] led us to define *Adaptive* Fuzzy Extractors (AFEs) that aim at recovering a stable key R even if readings drift naturally over time. In addition to primitives Gen and Rep, an update primitive Upd is introduced. The algorithm should output R even if the authentication value ω^i is not close to the enrolled ω as long as ω^i has drifted from ω and Upd has been run regularly. We propose a generic methodology to design an AFE out of a RPI and a nonreusable FE. Once again, an efficient instantiation for the set difference is proposed. We are aware of no previous study that considers the problem of key derivation from drifting data such as browser fingerprints.

4.1 Preliminaries

4.1.1 Notation

$GF(n)$ denotes the finite field of n elements. $x \leftarrow f(.)$ denotes that x is an output of a function f. If f is randomized, we use the semicolon to make the randomness explicit. $f(x;\mu)$ is the result of f computed on x with randomness μ.

Let H denotes a cryptographic hash function modeled as a random oracle $\mathsf{H} : \{0,1\}^* \times \{0,1\}^{l_1} \rightarrow \{0,1\}^{\kappa}$ For any entity $\mathcal{E}$, we denote by $\mathcal{E}(z)$ the fact that $\mathcal{E}$ has knowledge of z. U_l denotes the uniformly distributed random variable on $\{0,1\}^l$. For a distinguisher D (or a class of distinguishers $\mathcal{D}$), we write the computational distance between X and Y as $\delta^D(X,Y) = |\mathbb{E}[D(X)] - \mathbb{E}[D(Y)]|$. When X and Y indistinguishable given security parameter sec ($\delta^D(X,Y) \leqslant \epsilon_{\text{sec}}$), we denote $X \approx Y$. $\mathcal{D}_{s_{\text{sec}}}$ denotes the class of randomized circuits which output a single bit and have site at most s_{sec}. We say $W^1, \ldots, W^n$ random variables are *correlated by subset* if it exists $p \leqslant n$ such as $\{i_1, \ldots, i_p\} \subset \{1, \ldots, n\}$ and $W^{i_1}, \ldots, W^{i_p}$ are correlated. Let λ denotes a security parameter. Except stated otherwise, we have $l = l(\lambda)$, $\kappa = \kappa(\lambda)$, $m = m(\lambda)$,$m_1 = m_1(\lambda)$, $m_2 = m_2(\lambda)$, $s_{\text{sec}} = \mathtt{poly}(\lambda)$ and $\epsilon_{\text{sec}} = \mathtt{ngl}(\lambda)$.

4.1.2 Background and Tools

Set Difference Metric

Let $\mathcal{M}$ consists in all subsets of a universe $\mathcal{U}$. For two sets ω and ω' belonging to $\mathcal{M}$, their symmetric difference is defined as $\omega \Delta \omega' \stackrel{def}{=} \{x \in \omega \cup \omega' | x \notin \omega \cap \omega'\}$. The set difference metric between ω and ω' is the defined as $d(\omega, \omega') \stackrel{def}{=} |\omega \Delta \omega'|$. Recall the bin-set equivalence: if ω denotes a set, it can be viewed a binary vector in $\{0,1\}^n$, with 1 at position x if $x \in w$, and 0 otherwise. Viewed in this way, set difference can be expressed

as Hamming distance between these two vectors. This transform is not applicable when the universe size n is super-polynomial.

Entropy Notions

Entropy specifies the amount of information contained in some data. In security-related contexts, we care about the (non) ability (for an adversary) to guess the value of a random variable. In the information-theoretic case, we often rely on the well-suited notion of *min-entropy*. A random variable A has min-entropy m, denoted $H_\infty(A) = \mathrm{m}$, if A has predictability 2^{-m} *i.e.* $\max_a Pr[A = a] = 2^{-m}$. Put another way, we have $H_\infty(A) \stackrel{def}{=} -\log(\max_{a \in A} P[A = a])$. The *average* min-entropy of A given B is

$$\tilde{H}_\infty(\mathrm{A}|\mathrm{B}) = -\log(\mathbb{E}_{b \in \mathrm{B}} \max_a \Pr[\mathrm{A} = a|\mathrm{B} = b]).$$

HILL entropy is a commonly used computational notion of entropy [158]. It was extended to the conditional case by Hsiao, Lu, and Reyzin [159].

Definition 1. *Let (W, S) be a pair of random variables. W has* HILL entropy *at least k conditioned on S, denoted $H^{\mathtt{HILL}}_{\epsilon, s_{sec}}(W|S) \geq k$ if there exists a collection of distributions X_s giving rise to joint distribution (X, S), such that $\tilde{H}_\infty(X|S) \geq k$ and $\delta^{\mathcal{D}_{s_{sec}}}((W, S), (X, S)) \leq \epsilon$.*

Fuzzy Extractors

For the sake of clarity, we make some dedicated recalls about FEs

Fuzzy Extractors The original definition of FEs, due to Dodis *et al.* [13], was information theory-based. We focus on the computational definition introduced in [83]. Fuller *et al.* extend their definition to any family of distributions and we adopt this convention.

Definition 2 (Fuzzy Extractor)**.** *A pair of randomized procedures "generate" (Gen) and "reproduce" (Rep) is a $(\mathcal{M}, \mathcal{W}, l, t)$-computational fuzzy extractor that is (ϵ, s_{sec})-hard if Gen and Rep satisfy the following properties:*

- *The generate procedure Gen on input $\omega \in \mathcal{M}$ outputs an extracted string $R \in \{0,1\}^l$ and a helper string $P \in \{0,1\}^*$.*
- *The reproduction procedure Rep takes an element $\omega' \in \mathcal{M}$ and a bit string $P \in \{0,1\}^*$ as inputs.*

- *The correctness property guarantees that if* $d(\omega, \omega') \leqslant t$ *and* $(R, P) \leftarrow \mathsf{Gen}(\omega)$*, then* $\mathsf{Rep}(\omega', P) = R$*. If* $d(\omega, \omega') > t$*, then no guarantee is provided about the output of* Rep*.*
- *The security property guarantees that for any distribution* $W \in \mathcal{W}$ *on* $\mathcal{M}$*, the string* R *is pseudorandom conditioned on* P *i.e.* $\delta^{s_{sec}}((R, P), (U_l, P)) \leqslant \epsilon$*.*

Dodis *et al.* also define *average-case* FEs for which the security property requires that for any auxiliary variable I, $((R, P, I), (U_l, P, I))$ appear indistinguishable. We also refer to fuzzy extractors that are secure for all distributions of (average) min-entropy m, in this case we replace $\mathcal{W}$ with the parameter m.

Reusable Fuzzy Extractor Reusability of fuzzy extractors [85] can be stated as the possibility to call procedure Gen numerous times on the noisy readings of ω while retaining security. Let us consider ρ readings $\omega^1, \ldots, \omega^\rho$ of the same fuzzy secret from which the user will be enrolled on ρ different authentication servers. Gen will then generate ρ couples $(R^1, P^1), \ldots, (R^\rho, P^\rho)$ where $(R^i, P^I) \leftarrow \mathsf{Gen}(w^i)$. Recalling that P^j's are meant to be public and that different servers should not trust each other, Canetti *et al.* [94] proposed a security model for which a given R^{i_0} is secure even if all the R^js (for $j \neq i_0$) are given to an adversary.

Definition 3 (Reusable Fuzzy Extractor [94])**.** *Let* $(\mathsf{Gen}, \mathsf{Rep})$ *be a* $(\mathcal{M}, \mathcal{W}, l, t, \epsilon)$*-FE and* $W^1, W^2,$
$\ldots, W^\rho$ *be* ρ *correlated random variables over* $\mathcal{M}$ *where* $W^i \in \mathcal{W}$ *for all* $1 \leq i \leq \rho$*. Let* D *be an adversary. Define the following game for all* $j = 1, \ldots, \rho$*:*

- ***Sampling*** *The challenger* $\mathcal{C}$ *samples* $\omega^j \leftarrow W^j$ *for all* j *and* $\eta \leftarrow \{0,1\}^l$*.*
- ***Generation*** $\mathcal{C}$ *computes* $(R^j, P^j) \leftarrow \mathsf{Gen}(\omega^j)$ *for all* j*.*
- ***Distinguishing*** *The advantage of* D *consists in:*

$$Adv(D) \stackrel{def}{=} Pr[D(R^1, \ldots, R^\rho, \{P^j\}_{1 \leqslant j \leqslant \rho}) = 1]$$
$$- Pr[D(R^1, \ldots, R^{j-1}, \eta, R^{j+1}, \ldots, R^\rho, \{P^j\}_{1 \leqslant j \leqslant \rho}) = 1]$$

$(\mathsf{Gen}, \mathsf{Rep})$ *is* $(\epsilon_{sec}, \rho, s_{sec})$*-reusable if for all* $D \in \mathcal{D}_{s_{sec}}$ *and for all* $j = 1, \ldots, \rho$*,* $Adv(D) \leq \epsilon_{sec}$*.*

Tools

In the work of Canetti *et al.* [94], the secret key R outputted by Gen consists in a randomly generated value while the helper string P contains numerous ciphertexts of R encrypted

under substrings of the noisy secret ω. To securely encrypt numerous times the key R, they made use of digital lockers that are reusable symmetric encryption schemes with correlated and weak keys. The reusability of digital lockers implies the FE reusability. We now describe the tools demanded to design reusable FEs fitting our methodology.

Symmetric encryption scheme We will also require the use of a symmetric encryption scheme as introduced in Definition 28 that will be denoted (Enc, Dec). We require this encryption scheme to fulfill the "find-then-guess" chosen plaintext attack (FTG-CPA) security due to Bellare *et al.* [160] recalled in Definition 28. This notion is the closest notion to the classic CPA security defined for public key schemes.

Pseudoentropic Isometries We propose a new paradigm for designing reusable FEs. Given a fuzzy secret ω, a pseudoentropic isometry aims at randomly deriving Ω that retains the entropy of ω and the distances between inputs (at least locally). More precisely, a pseudoentropic isometriy consists in two randomized procedures (DerGen, DerRep) defined as follows.

Definition 4 (Pseudoentropic isometry)**.** *Let $(\mathcal{M}_1, d_1)$ and $(\mathcal{M}_2, d_2)$ be two metric spaces. A $(\mathcal{M}_1, \mathcal{M}_2, \mathcal{W}, m_2, \epsilon_{sec}, s_{sec})$-pseudoentropic isometry is a pair of randomized procedures (DerGen, DerRep) with the following properties:*

1. *DerGen on input $\omega \in \mathcal{M}_1$ outputs $\Omega \in \mathcal{M}_2$ and some $F \in \{0,1\}^*$.*
2. *DerRep takes an element $\omega' \in \mathcal{M}_1$ and a bit string $F \in \{0,1\}^*$ as inputs to output $\Omega' \in \mathcal{M}_2$. The correctness property guarantees that if $(\Omega, F) \leftarrow \mathsf{DerGen}(\omega)$, then $d_2(\Omega, \Omega') \leq d_1(\omega, \omega')$. Else, no guarantee is provided about Ω'.*
3. *Let W a distribution, DerGen(W) is denoted (U, V). If $\omega \xleftarrow{\$} W$ and $(\Omega, F) = \mathsf{DerGen}(\omega)$, we denote $(\Omega, F) \xleftarrow{\$} (U, V)$. The security property guarantees that for any distribution $W \in \mathcal{W}$, we have $H^{\mathtt{HILL}}_{\epsilon_{sec}, s_{sec}}(W|V) \geqslant m_2$ and $H^{\mathtt{HILL}}_{\epsilon_{sec}, s_{sec}}(U|V) \geqslant m_2$.*

This notion is related to biometric embeddings used in [13]. A biometric embedding projects any fingerprint value into a metric space where a FE exists while loosely maintaining distances. On their own pseudoentropic isometries are not novel (the identity function is a pseudoentropic isometry). A reusable pseudoentropic isometry or RPI is the key to our approach. In an RPI, the knowledge of previous derived values does not help D to distinguish a random value from a newly derived projection obtained via DerGen. Drawing on the definition of reusability for FEs (Definition 3), we define a RPI as follows.

Definition 5 (RPI)**.** *Let $W^* \in \mathcal{W}$ be a fixed distribution. Let $W^1, W^2, \ldots, W^\rho$ be ρ correlated random variables over $\mathcal{M}_1$. Let D an adversary. Using notation of Definition 4, we define the following game for all $j = 1, \ldots, \rho$:*

- ***Sampling*** *The challenger* $\mathcal{C}$ *jointly samples* $\omega^j \leftarrow W^j$. *Then independently samples* $\omega^* \xleftarrow{\$} W^*$.
- ***Generation*** $\mathcal{C}$ *generates* $(\Omega^j, F^j) \leftarrow \mathsf{DerGen}(\omega^j)$ *and* $(\Omega^*, F^*) \leftarrow \mathsf{DerGen}(\omega^*)$.
- ***Distinguishing*** *The advantage of* D *consists in:*

$$\begin{aligned} Adv(D) \stackrel{def}{=} & Pr[D(\Omega^1, \ldots, \Omega^\rho, F^1, \ldots, F^\rho) = 1] \\ & - Pr[D(\Omega^1, \ldots, \Omega^{j-1}, \Omega^*, \Omega^{j+1}, \ldots, \Omega^\rho, F^1, \ldots, F^\rho) = 1] \end{aligned}$$

(DerGen*,* DerRep*) is said to be* ρ*-reusable if for all* $D \in \mathcal{D}_{s_{sec}}$ *and for all* $j = 1, \ldots, \rho$*, the advantage* $Adv(D) \leq \epsilon_{sec}$.

4.2 From nonreusable to reusable Fuzzy Extractors

Before the recent work of Canetti *et al.*, all Hamming-based [90] and set difference-based [92] constructions, were prone to reusability issue. In this section, we introduce a new and generic way to address reusability. The idea is to first use a RPI to randomize fuzzy secrets and then apply a nonreusable FE on unrelated projected values. Based on nonreusable FEs due to Dodis *et al.* [13], we propose a reusable FE that enjoys satisfying storage, time and correction complexities.

4.2.1 High Level Overview

Let $(\mathsf{Gen}', \mathsf{Rep}')$ denote a (*average-case*) nonreusable FE. Taking as input a fuzzy secret ω, the generation procedure Gen' implicitly draws a ball $\mathcal{B}(\omega, t)$ centered in ω where the radius t consists in the error tolerance of the fuzzy extractor. Whenever a noisy reading ω' is given to procedure Rep', the secret key will be recovered as long as ω' belongs to $\mathcal{B}(\omega, t)$. In most cases and notably for constructions based on secure sketches, if numerous helper strings $P^1_\omega, \ldots, P^\rho_\omega$ are generated (Gen') from the same fuzzy secret $\in \mathcal{B}(\omega, t)$, all the P^is inherently contain information about ω which leads to reusability issue.

Let us consider the situation where a user wants to enroll its fuzzy secret ω on ρ authentication servers. To address reusability, the key idea is to randomly project the ρ fuzzy versions of ω onto unrelated values. By using an adapted ρ-RPI, the user gets unrelated values $\Omega_1, \ldots, \Omega_\rho$ that will be each enrolled once, respectively toward servers $1, \ldots, \rho$. Now whenever she wants to authenticate herself toward server j from ω', the user uses the aforesaid RPI to get Ω'^j verifying $d(\Omega'^j, \Omega'^j) = d(w, w')$. Since the Ω^js are -a priori- uncorrelated while each Ω^j undergoes only one generation procedure. Hence,

one should be able to enroll herself from the same fuzzy secret without fearing reusability issue. This idea is summed up in Figure 4.2.1.

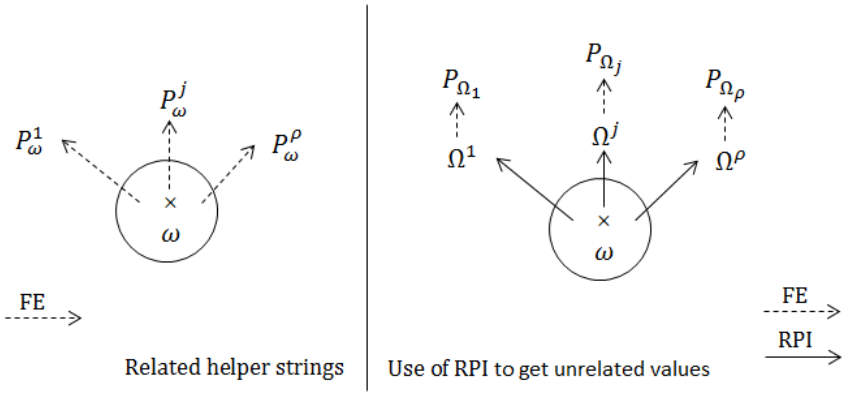

Figure 4.1: Overview of reusability via RPI randomization

4.2.2 Generic methodology

In this subsection, we detail our generic methodology to design a reusable FE (Gen, Rep) out of a nonreusable one and an adapted RPI.

Let (DerGen, DerRep) be a ρ-RPI from $\mathcal{M}_1$ to $\mathcal{M}_2$. Let (Gen', Rep') be an average-case FE over $\mathcal{M}_2$ correcting t errors. The generation procedure Gen will first call DerGen to randomize the input ω into Ω. The nonreusable FE is then applied on Ω. The RPI ensures that $d_2(\Omega, \Omega') \leq d_1(\omega, \omega')$ while the correctness of the underlying nonreusable FE ensures that Rep' recovers R from Ω' and the associated helper string as long as $d_2(\Omega, \Omega') \leqslant t$. Overall this leads to recovering R as long as $d_1(\omega, \omega') \leqslant t$.

In addition, the RPI ensures that different generated helper strings that came from unrelated randomized values ensure reusability assuming security of (Gen', Rep') in the single use case. One should notice that this methodology also applies to secure sketches.

Generation procedure Gen
Input: $w \in \mathcal{M}_1$.
1. $(\Omega, F) \leftarrow \mathsf{DerGen}(w)$.
2. $(R, Q) \leftarrow \mathsf{Gen'}(\Omega)$.
3. Set $P = (F, Q)$.
4. Return (R, P).

Reproduction procedure Rep
Inputs: $w' \in \mathcal{M}_1$,
Helper data $P \in \{0,1\}^*$.
1. Parse $P = (F, Q)$
2. $\Omega' \leftarrow \mathsf{DerRep}(w', F)$.
3. $R \leftarrow \mathsf{Rep'}(\Omega', Q)$.
4. Return R.

Figure 4.2: A generic reusable FE

Note: Even in the nonreusable setting this approach has benefits. For distributions W where the number of possible error patterns is larger than the $H_\infty(W)$, the bounds of

traditional fuzzy extractors provide little security (see [13, Theorem 5.1] and the discussion in [94]). However, it may be possible to use a pseudoentropic isometric to avoid these bounds and provide better parameters.

Security Analysis Correctness and security properties rely on those of (DerGen, DerRep) and (Gen',Rep').

Theorem 1. *Let (DerGen, DerRep) be a* $(\mathcal{M}_1, \mathcal{M}_2, \mathcal{W}, m_2, \epsilon_{RPI}, s_{RPI})$*-RPI that is* ρ*-reusable and (Gen', Rep') be an average-case* $(\mathcal{M}_2, m_2, l, t)$*-FE that is* (ϵ_{FE}, s_{FE})*-hard. Then Figure 4.2 defines a* $(\mathcal{M}_1, \mathcal{W}, l, t)$*-FE that is* $(\epsilon_{sec}, \rho, s_{sec})$*-reusable for* $\epsilon_{sec} = 4\epsilon_{RPI} + \epsilon_{FE}$ *and* $s_{sec} = \min\{s_{RPI} - |\mathsf{Gen}'|, s_{FE}\}$.

Proof. The correctness is straightforward and follows from aforesaid explanations. To ensure security, we first show that R appears pseudorandom even in presence of P and then treat reusability.

Under notation of Definition 4, we have that Ω and F respectively come from distribution U and V such as $H^{\mathtt{HILL}}_{\epsilon_{\mathrm{RPI}}, s_{\mathrm{RPI}}}(U|V) \geqslant m_2$. We first show that fuzzy extractors work on distributions with HILL entropy.

Lemme 2. *Let* U, V *be a joint distribution where* $H^{\mathtt{HILL}}_{\epsilon_{RPI}, s_{RPI}}(U|V) \geqslant m_2$ *and let (Gen', Rep') be an average-case* $(\mathcal{M}_2, m_2, l, t)$*-FE that is* (ϵ_{FE}, s_{FE})*-hard. Define* $R, P \leftarrow \mathsf{Gen}'(U)$*, then*

$$\delta^{\mathcal{D}_s}((R, P, V), (U_l, P, V)) \leqslant \epsilon.$$

for $\epsilon = 2\epsilon_{RPI} + \epsilon_{FE}$ *and* $s = \min\{s_{RPI}, s_{FE}\}$.

Proof. Suppose not, that is suppose that there exists some D of size at most s such that $\delta^D((R, P, V), (U_l, P, V)) > \epsilon$. Let X_v be a set of distributions giving rise to a joint distribution such that $\tilde{H}_\infty(X|V) \geq m_2$. Consider a D_1 that does the following:

1. Receive input α, β.
2. Run $\gamma, \nu \leftarrow \mathsf{Gen}'(\alpha)$.
3. Output $D(\gamma, \nu, \beta)$.

Also consider a D_2 that does the following:

1. Receive input α, β.
2. Run $\gamma, \nu \leftarrow \mathsf{Gen}'(\alpha)$.
3. Sample random string $u \leftarrow U_\ell$.
4. Output $D(u, \nu, \beta)$.

Denote $R', P' \leftarrow \mathsf{Gen}'(X)$. By the triangle inequality we have the following:

$$\begin{aligned}
&\delta^{D_1}((U,V),(X,V)) + \delta^{D_2}((U,V),(X,V)) \\
&\quad = \delta^{D}((R,P,V),(R',P',V)) + \delta^{D}((U_\ell,P,V),(U_\ell,P',V)) \\
&\quad \geq \delta^{D}((R,P,V),(U_\ell,P,V)) - \delta^{D}((U_\ell,P',V),(R',P',V)) \\
&\quad \geq \epsilon - \epsilon_{\mathrm{FE}} = 2\epsilon_{\mathrm{RPI}}
\end{aligned}$$

Thus, either D_1 or D_2 distinguishes U, V from X, V with advantage at least ϵ_{RPI}. Either of these distinguishers contradict the HILL entropy of U, V. This completes the proof. □

Lemma 2 allows us to conclude that $\delta^{\mathcal{D}_s}((R,Q,F),(U_l,Q,F)) \leqslant \epsilon$. That is,

$$\delta^{\mathcal{D}_s}((R,P),(U_l,P)) \leqslant \epsilon$$

for $P = (F,Q)$, and aforesaid parameters $\epsilon = 2\epsilon_{\mathrm{RPI}} + \epsilon_{\mathrm{FE}}$, $s = \min\{s_{\mathrm{RPI}}, s_{\mathrm{FE}}\}$.

Reusability Let $W^1, \ldots, W^\rho$ be correlated distributions over $\mathcal{M}_1$, where $W^i \in \mathcal{W}$ for all j. The following games consist in a challenger $\mathcal{C}$ trying to fool D for some distinguished i_0:

$\mathcal{G}_0$ $\mathcal{C}$ honestly samples values as prescribed in Definition 3 and sends

$$(R^1, F^1, Q^1), \ldots, (R^{i_0}, F^{i_0}, Q^{i_0}), \ldots, (R^\rho, F^\rho, Q^\rho)$$

to D.

$\mathcal{G}_1$ In this game, there is one change compared to the previous one. $\mathcal{C}$:

1. Samples the ω^js and then uses DerGen to obtain $(\Omega^1, F^1), \ldots, (\Omega^\rho, F^\rho)$.
2. Replaces ω^{i_0} with random $\omega^* \leftarrow W^*$ (where W^* as prescribed in Definition 5).
3. Computes $(\Omega^*, F^*) \leftarrow \mathsf{DerGen}(\omega^*)$ and $(R^*, Q^*) \leftarrow \mathsf{Gen}'(\Omega^*)$
4. Sets $P^* = (F^{i_0}, Q^*)$
5. Gives D the actual R^js and P^js except for $j = i_0$ for which he receives (R^*, P^*).

If D can distinguish this game from the previous one, he would then be able to distinguish the distribution with Ω^{i_0} from the one with Ω^*. This breaks the reusability of the RPI. That is, $\mathcal{G}_1$ appears indistinguishable from $\mathcal{G}_0$ for $\epsilon = \epsilon_{\mathrm{RPI}}$ and $s = s_{\mathrm{RPI}} - |\mathsf{Gen}'|$.

$\mathcal{G}_2$ In this game, after computing $(R^*, Q^*) \leftarrow \mathsf{Gen}'(\Omega^*)$, $\mathcal{C}$ discards the value R^* and replaces it with some $\eta \xleftarrow{\$} \{0,1\}^l$ randomly sampled. Since $H^{\mathtt{HILL}}_{\epsilon_{\mathrm{RPI}}, s_{\mathrm{RPI}}}(\Omega^*|F^*) \geq m_2$ then $H^{\mathtt{HILL}}_{\epsilon_{\mathrm{RPI}}, s_{\mathrm{RPI}}}(\Omega^*|F^{i_0}) \geq m_2$. Thus by Lemma 2, (U_l, P^*) and (R^*, P^*) are computationally indistinguishable. Hence, this game is indistinguishable from the previous one for $\epsilon = 2\epsilon_{\mathrm{RPI}} + \epsilon_{\mathrm{FE}}$ and $s = \min\{s_{\mathrm{RPI}}, s_{\mathrm{FE}}\}$.

$\mathcal{G}_3$ In the previous game, D was given $(R^1, F^1, Q^1), \ldots, (\eta, F^{i_0}, Q^*), \ldots, (R^\rho, F^\rho, Q^\rho)$ where η is random and does not depend on P^*. In this game, $\mathcal{C}$ sends the actual Q^{i_0} (obtained via computed $\mathsf{Gen}'(\Omega^{i_0})$ instead of Q^*.

If D can distinguish that Q^{i_0} has been given instead of Q^* (obtained via computed $\mathsf{Gen}'(\Omega^*)$, he can in particular distinguish Ω^{i_0} from Ω^*. Hence, he can distinguish

$$(\Omega^1, \ldots, \Omega^{i_0}, \ldots, \Omega^\rho, F^1, \ldots, F^{i_0}, \ldots, F^\rho)$$

from

$$(\Omega^1, \ldots, \Omega^{i_0-1}, \Omega^*, \Omega^{i_0+1}, \ldots, \Omega^\rho, F^1, \ldots, F^{i_0}, \ldots, F^\rho).$$

This contradicts the reusability of the RPI. Thus, $\mathcal{G}_3$ is indistinguishable from $\mathcal{G}_2$ for $\epsilon = \epsilon_{\mathrm{RPI}}$ and $s = s_{\mathrm{RPI}} - |\mathsf{Gen}'|$.

In $\mathcal{G}_3$, D is given $(R^1, P^1), \ldots, (\eta, P^{i_0}), \ldots, (R^\rho, P^\rho)$ where η is randomly sampled. By transitivity, this latter game is indistinguishable from $\mathcal{G}_0$. This latter indistinguishability is exactly the one required by Definition 3. □

4.2.3 A Set Difference-based RPI

In this subsection, we present a set difference-based RPI that will enable us to instantiate our generic methodology described in previous subsection. Canetti *et al.* recently designed a reusable FE based on the Hamming distance. Even if natural to study, this metric covers a few use cases (iris fingerprint) while the set difference metric appears well-fitted in numerous contexts. In addition to biometrics (*e.g.* digital fingerprint), this metric appears of particular interest to deal with device and browser fingerprints (*e.g.* list of applications, list of contacts, closest WiFi networks...).

Environment and Notation Set difference based fuzzy extractors presented in [13] take as inputs subsets of a universe $\mathcal{U}$ such as $n = |\mathcal{U}|$. We denote $(\mathcal{M}_{\mathcal{U}}, d)$, the metric space where $\mathcal{M}_{\mathcal{U}}$ consist in all the subsets of $\mathcal{U}$ and d is the set difference metric. Plus, $\mathcal{M}_{\mathcal{U},s}$ denotes the restriction of $\mathcal{M}_{\mathcal{U}}$ to s-elements subsets. In the following, $\mathcal{M}_\kappa$ denotes $(GF(2^\kappa), d)$ equipped with the set difference metric d. Similarly $\mathcal{M}_{\kappa,s}$ denotes the

restriction to sets of sizes s. Let W be a probability distribution over $\mathcal{U}$ with min-entropy m.

To design our set difference-based RPI, we will use the hash function $\mathsf{H} : \{0,1\}^* \times \{0,1\}^{l_1} \to \{0,1\}^{\kappa}$. When modeled as a non programmable random oracle, its outputs appear uniformly distributed and contain all the entropy of its input. In a nutshell, our set difference-based RPI, presented in Figure 4.3, consists in randomizing each set element through the use of H.

Algorithm DerGen
Input: $\omega = \{\omega_1, \ldots, \omega_s\}$,
$\forall 1 \leqslant i \leqslant s,\ \omega_i \in \mathcal{U}$.
1. $\mathsf{salt} \xleftarrow{\$} \{0,1\}^{l_1}$.
2. For $i = 1 \ldots s$,
   $x_i \leftarrow \mathsf{H}(\omega_i; \mathsf{salt})$.
3. Set $\Omega = \{x_1, \ldots, x_s\}$.
4. If $|\Omega| < s$,
   go to 1.
5. Return (salt, Ω).

Algorithm DerRep
Inputs: $\omega' = \{\omega'_1, \ldots, \omega'_s\}$,
$\forall 1 \leqslant i \leqslant s,\ \omega'_i \in \mathcal{M}$,
$\mathsf{salt} \in \{0,1\}^*$.
1. For $i = 1 \ldots s$,
   $x'_i \leftarrow \mathsf{H}(\omega'_i; \mathsf{salt})$.
2. Set $\Omega' = \{x'_1, \ldots, x'_{s'}\}$.
3. While $|\Omega'| < s$,
   $z \xleftarrow{\$} GF(2^{\kappa})$.
   $\Omega' \cup \{z\}$.
4. Return Ω'.

Figure 4.3: A set difference-based RPI

Step 4 of Algorithm DerGen aims at avoiding collision. In such a case, a new seed salt is chosen and the protocol starts again. Choosing κ big enough, one can be sure that the case with no collision occurs with overwhelming probability. In the same vein, Step 3 of Algorithm DerRep maintains distances between original values (w and w') and randomly derived ones (Ω and Ω'). Indeed, because of step 4 of DerGen, a collision occurring in DerRep can only be due to an element w'_i that did not appear in w. The condition we require on distributions is that each set has superlogarithmic min-entropy.

Proposition 1. *Let λ be the security parameter and fix $\epsilon > 0$. Define $\mathcal{W}$ as the set of all joint distributions W where for any i, $\tilde{H}_{\infty}(W^i | W^1, ..., W^{i-1}, W^{i+1}, ..., W^s) \geq k + 2\log(1/\epsilon) + O(1)$. Figure 4.3 defines a $(\mathcal{M}_{\mathcal{U},s}, \mathcal{M}_{\kappa}, \mathcal{W}, m_2, \epsilon, \infty)$-PI for the set difference metric where $m_2 = k \cdot s$ with an unbounded number of queries to the random oracle where $\epsilon' = \epsilon + (1 - e^{-s^2/2^k})$. In particular if $2^k = \omega(\mathtt{poly}(\lambda))$ then $\epsilon' = \epsilon + \mathtt{ngl}(\lambda)$.*

Proof. We have to prove both isometric and security properties.

1. *Isometry property.* By design, Ω is of size s and for any $w_i = w'_i$, then $x_i = x'_i$. Thus, $d(\Omega, \Omega') \leq d(w, w')$.

2. *Security.* Treating H modeled as a random oracle, first consider the case where the condition in step 4 is not triggered. Then H is a good extractor for each

individual enrollment. With H a random oracle, the knowledge of the random salt does not impact any entropy loss: $\forall\, \mathsf{salt}, H_\infty(X = x_i|V = \mathsf{salt}) = m$. That is, for an individual enrollment

$$(X_i, W_1, ..., W_{i-1}, W_{i+1}, ..., W_s, salt) \approx_\epsilon (U_k, W_1, .., W_{i-1}, W_{i+1}, ..., W_s, salt).$$

To show the statement of the proposition we measure the probability of the condition in step four being triggered.

$$\begin{aligned}
\Pr[\text{no collision}] &= \prod_{i=1}^{s}(1 - \frac{i}{2^k}) \\
&\geq \left(1 - \frac{s}{2^k}\right)^s \\
&= \left(\left(1 - \frac{s}{2^k}\right)^{2^k/s}\right)^{s^2/2^k} \\
&\geq (1/e)^{s^2/2^k} = e^{-s^2/2^k}
\end{aligned}$$

□

Adding reusability We now proceed to show reusability of the construction in this setting we need a slightly more restrictive definition of the random oracle and the set of sources $\mathcal{W}$. We need to ensure that the output length of the random oracle is longer enough that collisions occur with negligible probability and the entropy of each item is high enough that the adversary can query an input to the random oracle with only negligible probability. In the below theorem, we bound the size of an adversary by the number of random oracle queries it is permitted to make. In this theorem, the adversary may have unlimited computational power only the number of queries affects security.

Theorem 2. *Let λ be a security parameter, let $q = \mathtt{poly}(\lambda)$, $\rho = \mathtt{poly}(\lambda)$, $\kappa = \omega(\log \lambda), |\mathsf{salt}| = \omega(\log \lambda)$ and $m_2 = \omega(\log \lambda)$. Define $\mathcal{W}$ as the set of all joint distributions W where for any i, $\tilde{H}_\infty(W^i|W^{-i}) \geq \kappa$ where W^{-i} represents all other elements of W. Then Figure 4.3 defines a $(\mathcal{M}_{\mathcal{U},s}, \mathcal{M}_\kappa, \mathcal{W}, m_2, \epsilon, q)$-RPI for the set difference metric where $m_2 = \kappa \cdot s$ for $\epsilon = q(q+1)2^{-\kappa} + q2^{-m_2} = \mathtt{ngl}(\lambda)$.*

Notes: In the above theorem the running time of the adversary does not matter only the number of oracle queries. So instead of listing the distinguisher size we list the number of queries q they can make to the random oracle. Second, the proof only relies on an attacker finding a preimage to a random oracle output so the reusability ρ does not effect security parameters.

Proof. Fix some j that the adversary is trying to distinguish. The basic idea of the proof is that the adversary should have no information about the output of the random oracle on ω^j unless they have been able to find an input to the random oracle that matches its output.

Outputs of algorithms DerGen and DerRep consist in sets of random elements belonging to $GF(2^\kappa)$. By the use of H, these elements are uniformly distributed over $\mathcal{M}_\kappa$. Let $W^1, \dots W^\rho$ be related distributions from which $\omega^1, \dots, \omega^\rho$ are respectively sampled (some or all ω^js could then be equal). The adversary is given

$$(\mathsf{salt}^1, \Omega^1), \dots, (\mathsf{salt}^{j-1}, \Omega^{j-1}), (\mathsf{salt}^j, z), (\mathsf{salt}^{j+1}, \Omega^{j+1}), \dots, (\mathsf{salt}^\rho, \Omega^\rho)$$

where $(\mathsf{salt}^i, \Omega^i) \leftarrow \mathsf{DerGen}(\omega^j)$ and z is either Ω^j or the output of DerGen on an unrelated value.

The only way for an adversary D to learn about z is to find a input query $\omega^{i,*}$ such that $H(\omega^{i,*}, \mathsf{salt}^j) = z^i$ for some $1 \leq i \leq s$. Thus, we can consider strategies for generating input queries. Consider some enrollment value k. Each input symbol to the random oracle has input min-entropy $\omega(\log \lambda)$. Furthermore, while the random oracle returns a string with each query the only useful piece of information is whether the output value matches the provided output value in Ω^k. For convenience we assume that when an adversary receives a single matching value, they completely learn ω^*.

First note that the probability of a matching response on the first query is at most $\Pr[\text{match}] = \Pr[\text{correct point}] + \Pr[\text{collision}] \leq 2^{-\kappa} + 2^{-m_2}$. The probability of a collision does not change as D asks more queries but the probability of finding the correct point increases. For any $\omega_1, ..., \omega_q$ inputs to an oracle the adversary learns that at most one of these values matches (as they are assumed to win when an oracle input matches). Thus we have Let $A_1, A_2, \dots A_q$ be the random variables representing whether the ω_i was a correct value. Each A_i is just a bit, and at most one of them is equal to 1. Thus, the total number of possible responses is $q + 1$. Thus, we have the following,

$$\begin{aligned} \forall i, j, \tilde{H}_\infty(W^{i,j}|View(D(\cdot))) &= \tilde{H}_\infty(W^{i,j}|A_1, \dots, A_q) \\ &= H_\infty(W^{i,j}) - |A_1, \dots, A_q| \\ &= \kappa - \log(q+1)\,, \end{aligned}$$

where the second line follows from the first by [13, Lemma 2.2]. Thus, after all its queries the probability that of a match on the qth query is less than $2^{-\kappa - \log(q+1)} + 2^{-m_2}$. Thus, by union bound across all queries the total probability of a match is:

$$\Pr[\text{match}] \leq q(q+1)2^{-\kappa} + q2^{-m_2}.$$

This probability is negligible with the parameters as specified in the theorem. We then have the following:

$$
\begin{aligned}
Adv(D) =& \Pr[D(\Omega^1,\ldots,\Omega^\rho,F^1,\ldots,F^\rho)=1]\\
&-\Pr[D(\Omega^1,\ldots,\Omega^{j-1},\Omega^*,\Omega^{j+1},\ldots,\Omega^\rho,F^1,\ldots,F^\rho)=1]\\
&=\Pr[D(\Omega^1,\ldots,\Omega^\rho,F^1,\ldots,F^\rho)=1|\text{match}]\Pr[\text{match}]\\
&+\Pr[D(\Omega^1,\ldots,\Omega^\rho,F^1,\ldots,F^\rho)=1|\text{no match}]\Pr[\text{no match}]\\
&-(\Pr[D(\Omega^1,\ldots,\Omega^{j-1},\Omega^*,\Omega^{j+1},\ldots,\Omega^\rho,F^1,\ldots,F^\rho|\text{match})]\Pr[\text{match}]\\
&+\Pr[D(\Omega^1,\ldots,\Omega^{j-1},\Omega^*,\Omega^{j+1},\ldots,\Omega^\rho,F^1,\ldots,F^\rho|\text{no match})]\Pr[\text{no match}])\\
&=\Pr[\text{match}](\Pr[D(\Omega^1,\ldots,\Omega^\rho,F^1,\ldots,F^\rho)=1|\text{match}]\\
&-(\Pr[D(\Omega^1,\ldots,\Omega^{j-1},\Omega^*,\Omega^{j+1},\ldots,\Omega^\rho,F^1,\ldots,F^\rho|\text{match})]))\\
&+\Pr[\text{no match}](\Pr[D(\Omega^1,\ldots,\Omega^\rho,F^1,\ldots,F^\rho)=1|\text{no match}]\\
&-\Pr[D(\Omega^1,\ldots,\Omega^{j-1},\Omega^*,\Omega^{j+1},\ldots,\Omega^\rho,F^1,\ldots,F^\rho|\text{no match})])\\
&\leq \left(q(q+1)2^{-\kappa}+q2^{-m_2}\right)\cdot 1+(1-q(q+1)2^{-\kappa}+q2^{-m_2})\cdot 0\\
&=q(q+1)2^{-\kappa}+q2^{-m_2}.
\end{aligned}
$$

□

Corollary 1. *Using the RPI defined in Figure 4.3 for the family $\mathcal{W}$ defined above one can construct a reusable FE for any $s_{sec}=\texttt{poly}(\lambda),\rho=\texttt{poly}(\lambda)$ such that $\epsilon_{sec}=\texttt{ngl}(\lambda)$ and where $t=\Theta(n)$.*

Discussion Our instantiation of an RPI for the set difference metric (large universe) allows construction of the first reusable fuzzy extractor correcting a linear error rate that makes no assumption about how individual readings are correlated. The previous work of Boyen [85] assumed that the exclusive OR of two repeated enrollments leaked no information while the recent work of Canetti *et al.* [94] only achieves a sublinear error rate (and works for Hamming or set difference in the small universe setting).

Our construction also is very efficient due to the use of a nonreusable FE. Efficient information theoretic FEs are proposed in [13] for which storage and time complexities are respectively of order $t\log n$ (with equivalent entropy loss) and $\texttt{poly}(ns\log n)$. Reaping benefits of such constructions, our work then enjoys equivalent complexities.

In addition, we note that it is possible to view the second construction of Canetti *et al.* [94, Construction 2] as the composition of a RPI and non-reusable fuzzy extractor. Their construction is called *Lock-and-Error-Correct*. They first apply a digital locker to each dimension and then a fuzzy extractor. If the input source W has high entropy

in all dimensions, the digital locker serves as an RPI. Interestingly, the digital locker projects the set difference metric directly to the binary Hamming metric in contrast to our construction which remains in the large alphabet setting.

4.3 Adaptive Fuzzy Extractors

While Fuzzy Extractors were originally designed to handle any kind of noisy and keying material, they fast gained a lot of interest in the field of biometrics and PUFs. Unhappily as exhibited in Chapter 2, these applications suffer numerous limitations and in particular, biometrics still lack of practicability with respect to existing FEs. This state of affairs led us to consider the burgeoning alternative of browser and device fingerprints [110, 133, 111] that will naturally derive over time leading to changes much more important than genuine instability of classic biometrics. We then define *Adaptive* Fuzzy Extractors, meant to address this drifting problematic.

4.3.1 Environment: Alternative randomness and Drift problematic

Device and browser fingerprints (*e.g.* list of installed applications, list of cookies, ...) are much more versatile than biometrics. It raises the problematic of *uniquely* identifying users. More precisely, given a fingerprint value w', the goal is to decide if this fingerprint value is a new user or a previously encountered one that has undergone variations. In the latter case, the user has to be recognized and should not be assimilated to a new profile. Recent works have come with satisfying results with respect to this problematic [110, 133, 111]. Transposed in the context of authentication, this rises the problematic of authenticating users that might present numerous differences. Even if important, these differences do not prevent fingerprint values to be reliable [110, 133, 111]. In our context, this amounts to deal with a fuzzy extractor that can recover the actual key R even if the authentication value w' and the enrollment one w present more than t errors. The idea is to say that w' should have naturally drifted from w. Drawing a parallel with the *concept drift* problematic of predictive analytics, we refer to these fingerprint derivation as *fingerprint drift* and formalize it through the following definition

Definition 6. *Let $(\mathcal{M}, d)$ a metric space. Let $\omega^1, \ldots, \omega^\phi$ elements of $\mathcal{M}$ and an integer u. We say that $(\mathcal{M}, \omega^1, \omega^\phi, u)$ is a u-drift of length ϕ on $\mathcal{M}$ if for all $i = 1, \ldots, \phi - 1$, we have that $d(\omega^i, \omega^{i+1}) \leqslant u$.*

This definition is simple and may not capture all the subtleties a fingerprint may fit. This a first stone that should be improved at the light of further studies on fingerprinting.

In the context of fuzzy extractors, a naive answer to the fingerprint drift issue could be frequent re-enrollments. In practice, enrollment sessions always constitute critical sessions that organizations want to avoid. Plus, FEs were originally designed to enable the use of long term secrets. Frequent re-enrollment sessions would annihilate their primary goal.

4.3.2 High Level Overview

To cope with this drifting paradigm, we define *Adaptive* Fuzzy Extractors (AFEs) that enjoy a third primitive Upd compared to classic FEs. Without re-enrolling herself, a user should be able reproduce the same secret R than the one computed by Gen as long as variations between the enrollment value w and the authentication one w' follow an expected u-drift (Definition 6). A classic FE is meant to recover a previously extracted key R if and only if the reproduction value w' belongs to $\mathcal{B}(w,t)$. In our context, we require an AFE to recover the actual key R as long as the reproduction value w' has somewhat *naturally* drifted from w although w' does not belong $\mathcal{B}(w,t)$.

More precisely, given parameters $0 \leqslant u \leqslant t$, we propose AFEs to work according to two zones :

- *Updating Zone.* It can update the helper string value P before too many errors occur i.e. while w' is still close enough to w ($d(w,w') \leqslant u$).
- *Recovering Zone.* w' is close enough to w to enable key recovery but too far away to enable any helper string update ($u < d(w,w') \leqslant t$); further reproduction values w' should draw near w to enable an update.

We have that an adaptive fuzzy extractor can recover R as long as the fuzzy fingerprint w defines an u-drift (if an update is performed for each ω^i). Without representing updating zones for the sake of clarity, the philosophy of adaptive fuzzy extractors is depicted in Figure 4.3.2.

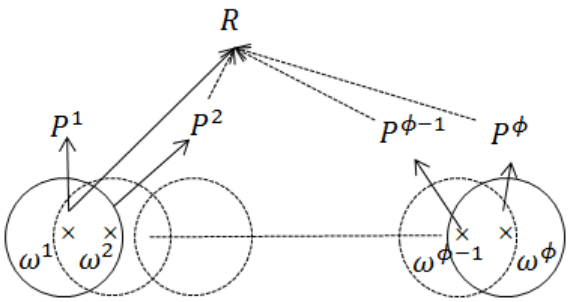

Figure 4.4: Philosophy of Adaptive Fuzzy Extractors.

4.3.3 Definition and Security Model

We now formally define Adaptive Fuzzy Extractors.

Definition 7 (Adaptive Fuzzy Extractor)**.** *A triple of randomized procedures "generate" (Gen), "update" (Upd), "reproduce" (Rep) is an* $(\mathcal{M}, \mathcal{W}, l, u, t, \phi)$*-adaptive fuzzy extractor that is* $(\epsilon_{sec}, s_{sec})$ *hard if the following holds:*

1. *On input* $\omega \in \mathcal{M}$*,* Gen *outputs an extracted string* $R \in \{0,1\}^l$ *and helper string* $P \in \{0,1\}^*$*.*
2. *On input* $\omega' \in \mathcal{M}$ *and* $P \in \{0,1\}^*$*, if either:*
   (a) $(R, P) \leftarrow \mathsf{Gen}(\omega)$ *and* $d(\omega, \omega') \leqslant u$*;*
   (b) *there exists* (ω^*, P^*) *such that* $P \leftarrow \mathsf{Upd}(\omega^*, P^*)$*,* $R \leftarrow \mathsf{Rep}(\omega^*, P^*)$ *and* $(\omega^*, \omega') \leqslant u$*,*

   then Upd *outputs an updated helper string* P'*.*
3. *The reproduction procedure* Rep *takes as inputs* $\omega' \in \mathcal{M}$ *and a bit string* $P \in \{0,1\}^*$*. The correctness guarantees that if either of conditions is satisfied:*
   (a) $(R, P) \leftarrow \mathsf{Gen}(\omega)$ *and* $d(\omega, \omega') \leqslant t$*;*
   (b) *there exists* (ω^*, P^*) *such that* $P \leftarrow \mathsf{Upd}(\omega^*, P^*)$*,* $R \leftarrow \mathsf{Rep}(\omega^*, P^*)$ *and* $(\omega^*, \omega') \leqslant t$*,*

   then $\mathsf{Rep}(\omega', P) = R$*. In any other case, no guarantee is provided about the output of* Rep*.*
4. *The security property guarantees that for correlated distributions* $W^1, \ldots, W^\phi$ *on* $\mathcal{M}$ *where* $W^i \in \mathcal{W}$*, the string* R *is pseudorandom even for those whose observe the* P^i*s for* $i = 1 \ldots \phi$*, generated through* Gen *or* Upd *from* $\omega^i \xleftarrow{\$} W^i$*. That is,*

$$\delta^{\mathcal{D}_{s_{sec}}}((R, \{P^i\}_{i=1}^{\phi}), (U_l, \{P^i\}_{i=1}^{\phi})) \leq \epsilon_{sec}.$$

Security Model

We now define reusability of AFEs. One can consider that the newly introduced drifting problematic brings another dimension to the security model. By taking the previous scenario where a user subscribes to ρ providers, this user should now be able to update ϕ times its profile on each server to handle variations. In such a context, helper strings generations (via Gen or Update) will be produced:

- when she subscribes to different services on a short period of time, her device fingerprint values are close and belong to correlated variables. This scenario reduces to the previous one (Definition 3);
- she may have to update her fingerprints toward a particular subscriber. With a fingerprint following a u-drift, the user may then generate helper strings from correlated and/or uncorrelated variables;
- when she subscribes to two different services at distant time periods for respective fingerprint values w_{i_0} and w_{i_1}, these values may appear as providing from uncorrelated variables.

Adapting reusability definition of [94] (Definition 3), we propose the following game-based definition for security of reusable AFEs.

Definition 8. *Let $(W^{1,1}, W^{2,1}, \ldots, W^{\phi,\rho})$ be potentially correlated random variables over $\mathcal{M}$ where each $W^{i,j} \in \mathcal{W}$. Let (Gen, Upd, Rep) be a $(\mathcal{M}, \mathcal{W}, \ell, u, t, \phi)$-AFE that is $(\epsilon_{sec}, s_{sec})$ hard. Let D an adversary. Let the following game for all $1 \leqslant j \leqslant \rho$:*

- ***Sampling*** *The challenger jointly samples $\omega^{1,k} \leftarrow W^{1,k}$ for $1 \leq k \leq \rho$, $\eta \in \{0,1\}^l$.*
- ***Helper string Generation and Drifting*** *The challenger computes helper strings via "generation" and "update" procedures.*
  - $(R^k, P^{1,k}) \leftarrow \mathsf{Gen}(\omega^{1,k})$
  - *for $2 \leqslant i \leqslant \phi$:* $P^{i,k} \leftarrow \mathsf{Upd}(\omega^{i,k}, P^{i-1,k})$.
- ***Distinguishing*** *The advantage of D is*

$$\begin{aligned} Adv(D) = &\Pr[D(R^1, \ldots, R^\rho, \{P^{i,k}\}_{i\leqslant\phi, k\leqslant\rho}) = 1] \\ &- \Pr[D(R^0, \ldots, R^{j-1}, \eta, R^{j+1}, \ldots, R^\rho, \{P^{i,k}\}_{i\leqslant\phi, k\leqslant\phi}) = 1] \end{aligned}$$

(Gen, Upd, Rep) is ρ-reusable if for all $D \in \mathcal{D}_{s_{sec}}$, the advantage is at most ϵ_{sec}.

Taking $\phi = 1$ leads to the reusability game of classic FEs (Definition 3) while taking $\rho = 1$ leads to the adaptive fuzzy extractor game (Definition 7).

4.4 From classic FE to Reusable AFE

To obtain a reusable AFE out of a classic FE, there are two key points to reach:

- computed helper strings should not leak information about user's fuzzy secret(s). As already seen in Section 4.2, RPIs address this point.

- a classic FE recovers an extracted R from w' as long as this latter is close enough to the enrollment value w. To continuously recover a key in spite of fingerprint derivation, the key idea is to generate a random stable key R that will be locked under keys with shorter lifespan. The temporary keys will be the ones outputted by a classic (*i.e.* non adaptive) FE.

4.4.1 Environment and Notation

Let (Enc, Dec) be a symmetric encryption scheme that is $(\epsilon_{\text{CPA}}, s_{\text{CPA}})$ FTG-CPA secure. Let (DerGen, DerRep) be a $(\mathcal{M}_1, \mathcal{M}_2, \mathcal{W}, m_2, \epsilon_{\text{RPI}}, s_{\text{RPI}})$-RPI that is $2\rho \cdot \phi$ reusable . Let $(\mathsf{Gen}_u, \mathsf{Rep}_u)$ be an average-case $(\mathcal{M}_2, m_2, l, u)$-FE that is hard $(\epsilon_{\text{FE}}, s_{\text{FE}})$. Let $(\mathsf{Gen}_t, \mathsf{Rep}_t)$ be an average-case $(\mathcal{M}_2, m_2, l, t)$-FE that is hard $(\epsilon_{\text{FE}}, s_{\text{FE}})$. Figure 4.5 depicts how to design a reusable AFE out of these tools.

4.4.2 Generation procedure

Given an enrollment value ω, the first step of the generation algorithm is to use the RPI to randomly project it onto two unrelated derived fingerprints Ω_u and Ω_t. Procedure Gen_u of the nonreusable FE correcting u errors will then be applied on Ω_u. Similarly, Gen_t is applied to Ω_t. Two temporary keys, K_u and K_t will be extracted. The first one will be used to detect if a fingerprint still belongs to the updating zone while the second one will be used to lock the randomly generated stable key R. In addition to helper strings, the overall helper string of our AFE contains encryptions of "1" and of R, respectively under K_u and K_t. string.

4.4.3 Update procedure

The update procedure takes as inputs a fuzzy version ω' and some helper data $P \in \{0,1\}^*$ to be updated into P'. The first step consists in deriving ω' into Ω'_u and Ω'_t. Algorithm Gen implicitly defined updating and recovering zones from Ω_u and Ω_t. Successful decryption of c_u under K_u recovers 1 and indicates that ω' is within distance u. It then makes sense to call update. If so, R can be unlocked by re-generating K_t. DerGen then randomizes ω' into Ψ_u and Ψ_t. Gen' computes new temporary keys K'_u and K'_t along with new helper strings. R is finally re-encrypted under K'_t.

4.4.4 Reproduction procedure

The reproduction procedure is straightforward. Taking as inputs ω' and some P, DerRep generates the corresponding Ω'_t as previously described. If Ω'_t is within distance t of the

Generation procedure Gen
Input: $\omega \in \mathcal{M}_1$.

1. *Derivation Step*
$(\Omega_u, F_u) \leftarrow \mathsf{DerGen}(\omega)$.
$(\Omega_t, F_t) \leftarrow \mathsf{DerGen}(\omega)$.

2. *Use of a classical FE*
$(K_u, Q_u) \leftarrow \mathsf{Gen'}(\Omega_u, u)$.
$(K_t, Q_t) \leftarrow \mathsf{Gen'}(\Omega_t, t)$.

3. *Key Generation*
$R \xleftarrow{\$} \{0,1\}^l$.

4. *Helper Data Generation*
$c_u = \mathsf{Enc}(K_u, 1)$,
$c_t = \mathsf{Enc}(K_t, R)$,

Set $P = (F_u, Q_u, c_u), (F_t, Q_t, c_t)$.
5. Return (R, P).

Update procedure Upd
Inputs: $\omega' \in \mathcal{M}_1$, $P \in \{0,1\}^*$.

1. *Fingerprint Check*
Parse $P = (F_u, Q_u, c_u), (F_t, Q_t, c_t)$.
$\Omega'_u \leftarrow \mathsf{DerRep}(\omega', F_u)$.
$\Omega'_t \leftarrow \mathsf{DerRep}(\omega', F_t)$.
$K_u \leftarrow \mathsf{Rep'}(\Omega'_u, Q_u)$.
$K_t \leftarrow \mathsf{Rep'}(\Omega'_t, Q_t)$.
$b \leftarrow \mathsf{Dec}(K_u, c_u)$
If b $\neq 1$, return $\perp$.
$R \leftarrow \mathsf{Dec}(K_t, c_t)$.

2. *Helper Data Re-generation*
$(\Psi_u, F'_u) \leftarrow \mathsf{DerGen}(\omega')$.
$(\Psi_t, F'_t) \leftarrow \mathsf{DerGen}(\omega')$.
$(K'_u, Q'_u) \leftarrow \mathsf{Gen'}(\Psi_u, u)$.
$(K'_t, Q'_t) \leftarrow \mathsf{Gen'}(\Psi_t, t)$.
$c'_u = \mathsf{Enc}(K'_u, 1)$, $c'_t = \mathsf{Enc}(K'_t, K)$.

3. Set $P' = (F'_u, Q'_u, c'_u), (F'_t, Q'_t, c'_t)$.
Return P'.

Reproduction Procedure Rep
Input: $\omega' \in \mathcal{M}_1$, $P \in \{0,1\}^*$.

1. *Parsing Helper Data*
Parse $P = (F_u, P_u, c_u), (F_t, P_t, c_t)$.
$\Omega'_t \leftarrow \mathsf{DerRep}(\omega', F_t)$.

2. *Key Reproduction*
$K_t \leftarrow \mathsf{Rep'}(\Omega'_t, P_t)$.
$R \leftarrow \mathsf{Dec}(K_t, c_t)$.

3. Return R.

Figure 4.5: Generation, Update, and Reproduce procedures

previously and implicitly enrolled Ω_t, then $\mathsf{Rep'}$ recovers K_t which enables to finally unlock R.

Theorem 3. *With notation defined in 4.4.1, Figure 4.5 defines a* $(\mathcal{M}_1, \mathcal{W}, \ell, u, t, \phi)$*-AFE that is* (ϵ_{FE}, s_{sec}) *hard and* ρ*-reusable for* $\epsilon = \phi(6\epsilon_{RPI} + \epsilon_{CPA} + 2\epsilon_{FE}), s_{sec} = \min\{s_{RPI} - 2(|\mathsf{Gen'}| + |\mathsf{Enc}|), s_{CPA}, s_{FE}\}$.

Remark 1. $2.\rho.\phi$ *reusability for a RPI means that there exists* $2.\rho.\phi$ *balls of radius* t *that lead (or have led during a certain period of time) to a successful authentication. Parameters have to be chosen so that such values remain very unlikely to be randomly predicted by any adversary. Recall that each of these balls is usually of exponential size in the distance parameter (either* u *or* t*).*

Proof. Correctness is straightforward. Once again, we separate pseudorandomness and reusability to deal with security. We begin by recalling that FTG-CPA security ensures that an adversary with an encryption oracle cannot distinguish between encryptions of two chosen messages.

Pseudorandomness As exhibited by following games, pseudrandomness of R comes from security of the encryption scheme (Enc,Dec).

$\mathcal{G}_0$ $\mathcal{C}$ samples $\omega \xleftarrow{\$} W$ where W is a distribution from $\mathcal{W}$. He then generates $(R, P) \leftarrow \mathsf{Gen}(w)$ as prescribed in Figure 4.5. $\mathcal{C}$ gives (R, P) to D.

$\mathcal{G}_1$ $\mathcal{C}$ instead of running DerGen on ω to get Ω_t runs on $\omega^* \leftarrow W^*$ (the distribution defined in Definition 5). Denote $(\Omega_t^*, F^*) \leftarrow \mathsf{DerGen}(\omega^*)$. The value Ω_t^* is substituted in the second Gen' process and the resulting encryption. This changes Q_t^* and c_t^*, all other values remain the same. By the reusability of the RPI, this game is indistinguishable from $\mathcal{G}_0$ for $\epsilon = \epsilon_{\mathrm{RPI}}$ and $s = s_{\mathrm{RPI}} - 2(|\mathsf{Gen}'| + |\mathsf{Enc}|)$.

$\mathcal{G}_2$ In the previous game, D was given R randomly sampled from $\{0,1\}^l$ and $P = (F_u, Q_u, c_u, F_t, Q_t^*, c_t^*)$. The only parts of P that are related to R are c_t^* and Q_t^* which are independent of the other values. Now, $\mathcal{C}$ samples some $\mu \xleftarrow{\$} \{0,1\}^l$ and computes $c^* = \mathsf{Enc}(K_t, \mu)$. $\mathcal{C}$ sets $P^* = (F_u, Q_u, c_u, F_t, Q_t^*, c^*)$ and sends (R, P^*). If D can distinguish this game from the previous one, he can in particular distinguish $(R, c_t^* = \mathsf{Enc}(K_t^*, R))$ from $(R, c^* = \mathsf{Enc}(K_t, \mu))$. First note by Lemma 2, K_t^* and K_t are both $2\epsilon_{\mathrm{RPI}} + \epsilon_{\mathrm{FE}}$ close to uniform. Hence, K_t and K_t^* are $2(2\epsilon_{\mathrm{RPI}} + \epsilon_{\mathrm{FE}})$ close. Finally, by FTG-CPA security of $(\mathsf{Enc}, \mathsf{Dec})$, $\mathcal{G}_2$ is indistinguishable from $\mathcal{G}_1$ for $\epsilon = 2(2\epsilon_{\mathrm{RPI}} + \epsilon_{\mathrm{FE}}) + \epsilon_{\mathrm{CPA}}$ and $s = \min\{s_{\mathrm{CPA}}, s_{\mathrm{RPI}}, s_{\mathrm{FE}}\}$.

$\mathcal{G}_3$ In the previous game, D is given R and $(F_u, Q_u, c_u, F_t, Q_t^*, c^*)$. Then, $\mathcal{C}$ can also sample some $\eta \xleftarrow{\$} \{0,1\}^l$ to finally give (η, P^*) to D. Since R and P^* are independent, this game is the same as the previous one to D's view.

$\mathcal{G}_4$ In the previous game, D was given η and $P^* = (F_u, Q_u, c_u, F_t, Q_t^*, c^*)$ that are independent. $\mathcal{C}$ can now replace c^*, Q_t^* with the actual values, independent of η, by the same reasoning as in $\mathcal{G}_1$. This indistinguishability holds for $\epsilon = \epsilon_{\mathrm{RPI}}$ and $s = s_{\mathrm{RPI}} - 2(|\mathsf{Gen}'| + |\mathsf{Enc}|)$

By transitivity, $\mathcal{G}_4$ is indistinguishable from $\mathcal{G}_0$ which leads to the indistinguishability required by Definition 7 for $\epsilon = 6\epsilon_{\mathrm{RPI}} + \epsilon_{\mathrm{CPA}} + 2\epsilon_{\mathrm{FE}}$ and $s = \min\{s_{\mathrm{RPI}} - 2(|\mathsf{Gen}'| + |\mathsf{Enc}|), s_{\mathrm{CPA}}, s_{\mathrm{FE}}\}$.

Reusability In the previous argument we first replaced the distribution Ω_t with an uncorrelated distribution, then replaced the ciphertext, the key R, and reverted back. Our strategy here is the same but it involves a hybrid argument where each update for a single enrollment has those values replaced. This leads to slightly worse parameters but the same overall structure.

Let $W^{1,1}, \ldots, W^{\phi,1}, W^{1,2}, \ldots, W^{\phi,\rho}$ be correlated distributions over $\mathcal{M}_1$, where $W^{i,k} \in \mathcal{W}$ for all i, k. Consider some fixed $1 \leq j \leq \rho$. The following games consists in a challenger $\mathcal{C}$ trying to fool D.

$\mathcal{G}_0$ $\mathcal{C}$ honestly samples values as prescribed in Definition 3 and sends

$$(R^1, P^{1,1}, \ldots, P^{\phi,1}), \ldots, (R^j, P^{1,j}, \ldots, P^{\phi,j}), \ldots, (R^\rho, P^{1,\rho}, \ldots, P^{\phi,\rho})$$

to D. Throughout this argument we will not modify R^i or $P^{k,i}$ for any $i \neq j$. Thus, we write this expression (reordering variables) as

$$(R^{-j}, P^{-j}, R^j, P^{1,j}, ..., P^{\phi,j}).$$

For all i, k, $P^{i,k}$ can be written $P^{i,k} = (F_u^{i,k}, Q_u^{i,k}, c_u^{i,k}, F_t^{i,k}, Q_t^{i,k}, c_t^{i,k})$ as specified in Figure 4.5.

$\mathcal{G}_1$ $\mathcal{C}$ instead of running DerGen on ω^j to get Ω_t^j runs on $\omega^* \leftarrow W^*$ (the distribution defined in Definition 5). Denote $(\Omega_t^{*,j}, F^{*,j}) \leftarrow \mathsf{DerGen}(\omega^*)$. The value $\Omega_t^{*,j}$ is substituted in the second Gen' process and the resulting encryption. This changes $Q_t^{*,j}$ and $c_t^{*,j}$, all other values remain the same. By the reusability of the RPI, this game is indistinguishable from $\mathcal{G}_0$ for $\epsilon = \epsilon_{\text{RPI}}$ and $s = s_{\text{RPI}} - 2(|\mathsf{Gen}'| + |\mathsf{Enc}|)$.

$\mathcal{G}_{2,...\phi}$ In each of these games we replace $\omega^{i,j}$ with independent samples from the distribution W^*. $\Omega_t^{i,j}$ is updated to $\Omega_t^{*,i,j}$ as in the game above. This replacement effects $Q_t^{*,i,j}$ and $c_t^{*,i,j}$, and not the other values. Each of these games is also indistinguishable for $\epsilon = \epsilon_{\text{RPI}}$ and $s = s_{\text{RPI}} - 2(|\mathsf{Gen}'| + |\mathsf{Enc}|)$.

$\mathcal{G}_{\phi+1,...,2\phi}$ At this point we replace the encrypted value $c^{*,i,j}$ one by one with random values. As in the pseudorandomness argument each game is indistinguishable for $\epsilon = \epsilon_{\text{CPA}} + 2(2\epsilon_{\text{RPI}} + \epsilon_{\text{FE}})$ and $s = \min\{s_{\text{CPA}}, s_{\text{RPI}}, s_{\text{FE}}\}$.

$\mathcal{G}_{2\phi+1}$ The key R is now replaced with a random η. Since R and the modified P are independent this game is statistically identical to the previous game.

$\mathcal{G}_{2\phi+1,...,3\phi+1}$ In each of these games a single pair of $Q_t^{*,i,j}$ and $c_t^{*,i,j}$ is replaced back with the actual values which are independent of η. Each game is indistinguishable from the previous for $\epsilon = \epsilon_{\text{RPI}}$ and $s = s_{\text{RPI}} - 2(|\mathsf{Gen}'| + |\mathsf{Enc}|)$.

By transitivity, this last game $\mathcal{G}_{3\phi+1}$ is indistinguishable from $\mathcal{G}_0$ for $\epsilon_{\text{sec}} = \phi(6\epsilon_{\text{RPI}} + \epsilon_{\text{CPA}} + 2\epsilon_{\text{FE}})$ and $s_{\text{sec}} = \min\{s_{\text{RPI}} - 2(|\mathsf{Gen}'| + |\mathsf{Enc}|), s_{\text{CPA}}, s_{\text{FE}}\}$ fulfilling Definition 8. □

Corollary 2. *Using the RPI defined in Figure 4.3 for the family $\mathcal{W}$ defined above one can construct a reusable and adaptive FE for any $s_{sec} = \mathtt{poly}(\lambda), \rho = \mathtt{poly}(\lambda), \phi = \mathtt{poly}(\lambda)$ such that $\epsilon_{sec} = \mathtt{ngl}(\lambda)$ where $t = \Theta(n)$.*

4.5 Conclusion et Perspectives

Dans ce chapitre, nous avons proposé le premier FE prouvé réutilisable pour la différence d'ensembles (dans le cas du *grand univers*) , résolvant un problème ouvert en 2004 par

Boyen [85] (résolution de L6). En particulier, notre instanciation constitue également le premier FE capable de corriger un taux d'erreurs linéaire tout en étant réutilisable et pourra se voir comme un complément des travaux de Canetti *et al.* En effet, rappelons que leur construction basée sur la métrique de Hamming peut être étendue à la différence d'ensemble dans le cas du *petit univers* mais jouit alors d'un taux de correction d'erreurs sous-linéaire. Tout comme leurs travaux, la sécurité de notre construction est assurée dans le modèle de l'oracle aléatoire.

En fait, notre FE basé sur la différence d'ensembles est une instanciation d'un framework générique dans lequel nous proposons de randomiser les secrets flous avant d'y appliquer un Fuzzy Extractors. Puisque les secrets des utilisateurs viennent potentiellement de distributions corrélées, l'idée est de les décorréler tout en maintenant les distances : nous avons alors introduit le concept de *Reusable Pseudoentropic Isometries* (RPIs) ou isometries pseudoentropiques réutilisables. Il est alors possible de concevoir un FE réutilisable à partir de tout FE classsique et d'une RPI adaptée.

En plus de nous concentrer sur la réutilisabilité, nous avons proposé le concept de Fuzzy Extractor Adaptatif ou *Adaptive Fuzzy Extractor* (AFE), particulièrement adapté au domaine du *fingerprinting* où des attributs –provenant d'un *grand univers*– souffrent d'une durée de vie plus courte. Un AFE a pour but de générer continuellement la même clé R même si les valeurs d'enrôlement w et d'authentification w' de l'attribut ont grandement varié et ce, à condition que w' ait *naturellement* dérivé w'.

Nous avons alors proposé une méthodologie générique permettant la conception d'un AFE à partir de tout FE réutilisable, également instanciée pour la différence d'ensembles particulièrement adaptée au domaine du *fingerprinting*. En effet, les attributs considérés sont principalement des listes (listes des contacts, applications, chansons, plug-ins,...) (résolution de L7). Notre modélisation via les *t-drift* est une première proposition. Une meilleure compréhension du comportement de ces attributs devra permettre une modélisation plus précise.

D'autre part, puisque l'usage d'une RPI permet également de décorréler des secrets, nous nous appuierons sur celle introduite à la Figure 4.3 pour nous affranchir de l'hypothèse **Hyp2**. En effet, les attributs considérés au chapitre précédent à la Section 3.6 sont des ensembles potentiellement corrélés. La RPI sera alors utilisée en amont pour assurer gloable du module FAST. Ce faisant, les valeurs d'attributs vont être décorrélés et apparaîtront indépendantes de sorte que nous pourrons nous ramener à l'hypothèse **Hyp2** (résolution de L8) pour prouver la sécurité du module FAST même si les attributs considérés sont corrélés.

La résolution de ces 3 limitations permet l'accession au stade TA3. L'accession à une solution industrialisable sera atteinte (stade TA3-Ind) lorsque les limitations L4 et L5 seront levées.

4.5.1 Perspectives

Pour les travaux futurs et une industrialisation du module FAST pour 2017, nous allons nous concentrer sur les limitations L4 et L5. Dans un premier temps, l'idée sera – à travers l'application App tel que décrite au stade TA1 – de remonter les valeurs d'attributs méconnus en les anonymisant pour étudier leur comportement (entropie, stabilité, modèle d'erreurs, ...). D'autre part, des travaux d'optimisation vont être lancés pour diminuer de moitié les temps de calcul rappelés à la Figure 3.11.

Limitation L9 ou la gestion du multi-devices

De manière orthogonale à ces travaux et comme rappelé au cours de l'introduction générale, l'Internet des Objets couplé à la menace de l'ordinateur quantique nous a conduit à étudier des schémas de signature de groupe post-quantum pour permettre aux utilisateurs d'être authentifié via n'importe lequel de ses objets connectés sans avoir à spécifier lequel. Déployer le module FAST sur chaque terminal de l'utilisateur ne permet pas d'assurer l'anonymat desdits terminaux. En effet, par conception, la clé retournée par FAST sk sera liée au terminal hôte. Ainsi, suivant une authentification TA (Figure 2.1), le serveur pourra distinguer par quel terminal l'utilisateur s'authentifie et ainsi apprendre des informations relatives à la vie privée de l'utilisateur. Par exemple, une authentification via tablette ou PC pourra induire que l'utilisateur est chez lui. Au contraire, l'utilisation du smartphone pourrait révéler qu'il est à l'extérieur. La fuite de ce genre de méta-information doit être évitée. Nous notons L9 cette limitation que les Chapitres 5 et 6 s'attacheront à résoudre.

Limitation	Définition
L9	Gestion du scénario multi-appareils avec anonymat de l'appareil impliqué.

Figure 4.6: Limitations du stade TA3

Chapitre 5 :

Stade TA4 - Signature de groupe en métrique de Hamming

Contents

Conformément aux jalons présentés lors de l'introduction générale, nous nous sommes aussi intéressés au domaine de la signature de groupe, vue comme une réponse possible au respect de la vie privée dans le paradigme de l'Internet des Objets et de l'avènement continu des réseaux sociaux. Un tel schéma consiste en un groupe d'utilisateurs capables de signer anonymement -et donc de s'authentifier- au nom de l'identité publique dudit groupe. Toute entité souhaitant vérifier la validité d'une signature pourra *seulement* vérifier que le signataire est effectivement un membre du groupe. Le manageur de groupe (*group manager*) fera figure d'autorité capable de lever l'anonymat des signatures en cas

de litige. On parle de schéma *dynamique* si une fois les paramètres publics générés, il est possible d'ajouter de nouveaux utilisateurs au cours de la vie du groupe [161].

Dans le contexte TA, authentifier un utilisateur $\mathcal{U}$ en ligne revient à s'assurer de son identité via un de ses appareils connectés alors que la croissance de l'IoT multiplie ce nombre d'appareils par utilisateur (ordinateur, smartphone, tablette, montre, ...). Á moyen terme, l'objectif de nos travaux sera de répartir l'application App sur ces tout ses terminaux pour lui permettre de s'authentififer à partir du terminal de son choix sans avoir à le spécifier. Puisqu'une authentification via son PC ou sa tablette signifiera que l'utilisateur était très probablement chez lui, ce genre de méta-données que le serveur pourrait apprendre constitue une première atteinte à notre leitmotiv qu'est le respect de la vie privée.

Pour y répondre, nous avons transposé la notion signature de groupe à l'ensemble des terminaux utilisateurs. Dans ce contexte, l'identité publique du groupe sera l'utilisateur alors que chacun de ses appareils pourra s'authentifier anonymement en son nom. D'autre part, certains des nos travaux R&D tendent à considérer le réseau social d'un utilisateur comme facteur d'authentification. Par exemple, la détection bluetooth d'un appareil appartenant à des relations de l'utilisateur contribuerait à l'authentifier. Dans ce cas le groupe associé à l'utilisateur serait l'ensemble des terminaux connus par l'utilisateur. Ainsi, le développement de l'IoT et l'avénement annoncé de l'ordinateur quantique nous a conduit à étudier des schémas de signature de groupe qui seraient résistants à l'ordinateur quantique. Dans notre contexte, un utilisateur ne sera pas amené à constituer des groupes de taille démentielle ; cependant, suivant la philosopie de TA et FAST, l'idée est que chaque terminal décide la valeur de sa clé de signature (ou au moins une partie de celle-ci) lorsqu'il est ajouté au groupe des appareils de l'utilisateur. Á l'époque des travaux, il n'existait aucun schéma de signature de groupe dynamique et résistant à l'ordinateur quantique.

Résultat d'une collaboration avec Olivier Blazy, Stéphane Cauchie et Philippe Gaborit, nous proposons la conception du premier schéma de signature de groupe basée sur la théorie des codes. S'appuyant sur une adaptation du protocole de Stern [162] converti en schéma de signature via le paradigme de Fiat-Shamir [47], ces travaux furent initialement été présentés à la conférence *Workshop on Coding and Cryptography 2015* [6]. Formalisant un nouveau type de preuve de connaissance, définie comme *Testable weak Zero-Knowledge*, la version longue de ces travaux est parue dans le journal *Design, Codes and Cryptography* [8]. La suite de ce chapitre, rédigée en anglais, reprend ces travaux.

5.1 Introduction

A group signature scheme allows members of a group to issue signatures on behalf of the group in an anonymous but revocable way: an opener is able to revoke anonymity of the actual signer in case of abuse. Since its introduction by Chaum and van Heyst [49], group signatures have been extensively studied. Bellare et al. [163] (BMW model) first gave formal security properties of group signature. Later, Bellare, Shi and Zhang [161] extended this model to dynamic groups (BSZ model). Numerous efficient group signatures such as [164, 165, 166] were proposed but only proven secure in a relaxation security of [163]. Delerablée and Pointcheval [167] proposed the first practical scheme fully fitting BSZ in the random oracle model (ROM) whereas Groth [168] also provided such a scheme but secure in the standard model. Then, as an improvement of group signatures, Kiayias, Tsiounis and Yung, suggested traceable signatures schemes in [169]. In addition to classic properties of a group signature scheme, a traceable signature enables the opening authority to delegate its revoking (or opening) capability to sub-openers but only against specific users. This gives two crucial advantages: sub-openers can run in parallel and authorities can monitor misbehaving users and then preserve honest users anonymity. The first efficient traceable signatures, provably secure in the standard model, were introduced by Libert and Yung in [170].

All these aforesaid schemes are pairing-based constructions. It was then worth looking for alternative since their security might collapse in front of quantum computers and that they involve heavy computations. Thus, many lattice-based constructions have been proposed such as [171] who first designed a lattice-based group signature scheme with both public key and signature size linear in the number of group members N. Recently, numerous works such as [172, 173, 174, 175] proposed more efficient lattice-based constructions where both sizes the group public keys and signatures are proportional to $log(N)$. In a concurrent and posterior work, Ezerman et al. [176] also designed a code based group signature that suffers weaker features in terms of size of parameters and properties. Plus, it is interesting to notice that, with the recent exception of [177], all lattice and code based constructions base their security on the static model of [163] meaning that our scheme constituted the first post-quantum dynamic group signature scheme.

Because of a restriction for adding new users (procedure Join), our scheme ensures security properties of traceability, anonymity and non-frameability in a relaxation of the BSZ model. Indeed, the security of our protocol is ensured only when an adversary can add honest users (oracle $\mathcal{O}^{\text{joinP}}$) that may be corrupted later while the BSZ model requires to fulfill security even in presence of an adversary adding already corrupted users: it led us to define our construction as *weakly dynamic*.

The main idea of our scheme consists in building an offset collision of two syndromes associated to two different matrices: a random one which enables to build a random syndrome from a *chosen* small weight vector; and a *trapdoor matrix*, which permits to find a small weight preimage of the previous random syndrome to which a fixed syndrome is added. These two small weight vectors will constitute the group member's secret signing key whose knowledge will be proved thanks to a variation of Stern's protocol.

Our contributions In this chapter, we propose a generic construction for designing the first code-based group signature tha we instantiate with the CFS scheme (see subsection 1.4). In a concurrent and independent work, Ezerman *et al.* [176] proposed a group signature scheme based on coding assumptions but only fitting the limited BMW model and with signatures and public key sizes linear in the number of group members.

Our security is based on a relaxation of the restrictive BSZ model with the properties of *anonymity*, *traceability* and *non-frameability*. Furthermore, it has numerous advantages over all existing post-quantum constructions and even some pairing based constructions: it allows to dynamically add new members (*weakly dynamic*) and enjoys nice features such as traceability in the sense of the KTY model and membership revocation. When instantiated with the CFS scheme [58], it leads to signatures and public keys sizes proportional to $N^{1/\sqrt{log(N)}}$ for N the number of group members, which is greater than a logarithmic complexity but asymptotically smaller than $N^{1/d}$ for any d. Plus, the proposed scheme can easily be extended into a traceable signature (according to the KTY model) handling membership revocation. In order to reach our goal, we introduce a new kind of proof of knowledge, referred as *Testable weak Zero Knowledge*. It can be seen as a weaker version of ZK proofs [55]: it allows a verifier to test whether a specific witness is used without learning anything more from the proof. This new notion, that could be of independent interest, appears particularly well-fitted in the context of group signature schemes. Under the Random Oracle Model (ROM), we ensure the security of our scheme by defining the *One More Syndrome Decoding* problem, a new code-based problem related to the Syndrome Decoding problem [57].

5.2 Preliminaries

In this section, we provide recalls necessary to the well understanding of this chapter.

5.2.1 Notation

μ still denotes some randomness. *crs* denotes the common string shared by involved parties in the Common Reference String (CRS) model [].

For a vector (resp. a string) v, $v[r]$ denotes the r-th coordinate (resp. symbol) of v. $\mathbb{F}_q$ denotes the finite field of cardinality q. $\mathcal{M}_{m\times n}(\mathbb{F}_q)$ denotes matrices over $\mathbb{F}_q$ of m rows and n columns. S_ω^n is the set of vectors of weight ω lying in $\mathbb{F}_2^n$ and Σ_n denotes the set of permutations over $[n]$. Here, $\omega t(x)$ denotes Hamming weight of x. λ still denotes a security parameter. For a protocol, l_λ denotes the number of iterations needed to reach the level of security required by λ.

H, $\mathsf{H}_\lambda : \mathbb{F}_2^* \longrightarrow \{0,1,2\}^{l_\lambda}$, $h : \mathbb{F}_2^* \longrightarrow \mathbb{F}_2^n$ and $h' : \mathbb{F}_2^* \longrightarrow \mathcal{M}_{k\times n}(\mathbb{F}_2)$ model random oracles.

For an instance of a group signature scheme, we denote by N the number of group members.

5.2.2 On code-based theory

One More Syndrome Decoding problem

Recalled in Chapter 1, the SD-problem is a problem based on coding theory and shown to be NP-complete [57] that consists in finding a small weight word for a given syndrome s. Let us notice that the case where one is asked to find a small solution where the weight is approximated within a constant was also proven to be computationally intractable in [178].

We now define a related problem, that we call the One-More Syndrome Decoding (OMSD) problem which we believe difficult. Let H a $(n-k)\times n$ random binary matrix, s a random syndrome and l vectors $x_1, x_2, \ldots, x_l$ of weight ω such as for any $i = 1 \ldots l$, $Hx_i^\top = s$.

Question Is it possible to find a $(l+1)$-th vector x_{l+1} such as $x_{l+1} \neq x_i$ for any $i = 1 \ldots l$, of weight ω and verifying $Hx_{l+1}^T = s$?

We denote $(H, s, \omega, \{x_i\}_{i=1\ldots l})$ such an OMSD instance where the goal is to find a valid x_{l+1}.

Discussion on the problem This problem corresponds to a code version of problems which are well known in number theory based cryptography [179]. In the case of coding theory and the Syndrome Decoding problem, the best known attacks are direct retrieval of independent syndromes, and it is widely believed that having an oracle which give you information on l independent syndromes does not help the attacker in finding a $(l+1)$-th solution to a syndrome decoding problem.

Stern's protocol For the sake of clarity, we recall Stern's protocol in Figure 5.1 previously introduced in Chapter 1.

Public data: $(H, s, \omega) \in \mathcal{M}_{(n-k)\times n}(\mathbb{F}_2) \times {\mathbb{F}_2}^{n-k} \times \mathbb{N}$
$\mathcal{P}$ proves knowledge of $z \in \mathbb{F}_2^n$ such as $Hz^\top = s$ and $\omega t(z) = \omega$

1. $\mathcal{P}$ computes commitments as follows:
   $\pi \xleftarrow{\$} \Sigma_n$, $u \xleftarrow{\$} \in \mathbb{F}_2^n$
   $c_1 = h(\pi || Hu^\top), c_2 = h(\pi(u)), c_3 = h(\pi(z+u))$
2. $\mathcal{P}$ sends c_1, c_2, c_3
3. $\mathcal{V}$ sends $\mathsf{ch} \xleftarrow{\$} \{0, 1, 2\}$
4. 3 cases for $\mathcal{P}$:
   $\mathsf{ch} = 0$: $\mathcal{P}$ sends π, u
   $\mathsf{ch} = 1$: $\mathcal{P}$ sends $\pi, z + u$
   $\mathsf{ch} = 2$: $\mathcal{P}$ sends $\pi(z), \pi(u)$
5. 3 cases $\mathcal{V}$:
   $\mathsf{ch} = 0$: $\mathcal{V}$ checks c_1, c_2
   $\mathsf{ch} = 1$: $\mathcal{V}$ checks c_1, c_3
   $\mathsf{ch} = 2$: $\mathcal{V}$ checks c_2, c_3
   and $\omega t(\pi(z)) = \omega$
6. $\mathcal{V}$ outputs Accept if all checks passed
   $\perp$ otherwise.

Figure 5.1: Stern's protocol

Remarks on Stern's protocol

The protocol presented here is the original one as introduced in [162]. It suffers a distinguishability issue since a verifier with knowledge of z is able to check whether this given witness z was actually involved during the protocol. Indeed, when $\mathsf{ch} = 0$ (respectively $\mathsf{ch} = 1$): in addition to checking c_1 and c_2 (resp. c_1 and c_3), such a verifier $\mathcal{V}(z)$ can also compute $c_3 = h(\pi(z + u))$ (resp. $c_2 = h(\pi(z + u + z))$ and then deduces involment of z. To fix this distinguishability issue, Stern then proposed a randomized version of its protocol for which commitments c_1, c_2, c_3 are computed by concatenating inputs to random seeds [67].

This ability to somewhat *trace* a user in the original version will constitute a central point for designing our group signature for which security will be ensured by defining what we call *Testable weak Zero-Knowledge* proofs (Definition 10).

Trapdoor Matrix

We propose a generic construction of a code-based group signature scheme through the use of what we call a *trapdoor matrix*. Such a matrix is actually hard to find and the only current candidate for instantiating our construction is the CFS matrix [58] (see Section 5.7).

Definition 9. *A trapdoor matrix family is a couple of polynomial algorithms* $(\mathsf{TrapGen}, \mathsf{Inv})$ *such that:*

- $\mathsf{TrapGen}(1^\lambda)$*: outputs a pair* $(Q, \mathsf{trk}) \in \mathcal{M}_{(n-k)\times n}(\mathbb{F}_2) \times \mathcal{TRK}$ *according to security parameter* λ*;*

- $\mathsf{Inv}(Q, \mathsf{trk}, s, \omega)$*: outputs, with non negligible probability, some* $x \in S_n^\omega$ *such as* $Qx^T = s$ *assuming* $(Q, trk) \leftarrow \mathsf{TrapGen}(1^\lambda)$*,* $s \in \mathbb{F}_2^{n-k}$ *and* $\omega \in \mathbb{N}$*.* x *has to appear random in* S_n^ω*. If no such solution is found, it returns* $\perp$*;*
- *(Correctness): for all* (Q, trk) *output by* $\mathsf{TrapGen}(1^\lambda)$*, and all* $s \in \mathbb{F}_2^{n-k}$*, we have that* $Q(\mathsf{Inv}(Q, trk, s, \omega))^\top = s$*;*
- *(One-wayness): for all polynomial adversary* $\mathcal{A}$*, the following is negligible:* $Pr[(Q, \mathsf{trk}) \leftarrow \mathsf{Gen}(1^\lambda); s \in \mathbb{F}_2^{n-k}; x \leftarrow \mathcal{A}(1^\lambda, Q, s, \omega) : (Qx^\top = s \text{ and } \omega t(x) = \omega)]$.

We say that Q is a *trapdoor matrix* and trk a *trapdoor key* if (Q, trk) was generated by $\mathsf{TrapGen}$.

5.3 Cryptographic Primitives Revisited

In this section, we first introduce a Stern-like protocol whose contribution could be of independent interest. Secondly, we define a new kind of proof of knowledge which aims at capturing the traceability issue of deterministic Stern-like protocols as highlighted above. After exhibiting how our new protocol can satisfy this new definition, we prove its security.

5.3.1 A variation on Stern's Protocol: Concatenated Stern's protocol

We propose a variation of Stern's protocol, referred as Concatenated Stern's protocol ($\mathcal{CSP}$) and depicted in Figure 5.2. The main idea consists in splitting the small weight secret z the prover will be challenged on and to run two related instances of Stern's protocol in parallel.

Let us consider the following SD-instance : $(H, s, 2\omega)$ for which z is said to be a *valid* solution if $z = (x\|y)$ where x and y have the same length and $\omega t(x) = \omega t(y) = \omega$. Then H can also be written as $H = (R\|Q)$ such as $Hz^\top = s \Leftrightarrow Rx^\top + Qy^\top = s$. We additionally assume that the verifier $\mathcal{V}$ can be given some y'. As exhibited at step 6 of Figure 5.2, he is then able to check in two out of three cases ($\mathsf{ch} = 0$ and $\mathsf{ch} = 1$) if $y' = y$ without learning x.

Compared to classic Stern's protocol, the interest of $\mathcal{CSP}$ is twofold: it enables to check independently the weight of each half of a small weight secret while a part of the secret can be compromised without revealing the entire secret. From now, by $\mathcal{CSP}$-instance, we refer to (H, s, ω) for which one is asked to find $z = (x\|y)$ such as $Hz^\top = s \wedge \omega t(x) = \omega t(y) = \omega$. Relying on Stern's protocol (Figure 5.1), we prove the security of $\mathcal{CSP}$ in subsection 3.4.

Public data: $(H, s, \omega) \in \mathcal{M}_{(n-k)\times 2n}(\mathbb{F}_2) \times \mathbb{F}_2^{n-k} \times \mathbb{N}$
$\mathcal{P}$ proves knowledge of a valid $z = (x\|y) \in \mathbb{F}_2^{2n}$ to $\mathcal{V}(y')$

1. $\mathcal{P}$ computes commitments as follows:
$(\pi, \sigma) \xleftarrow{\$} \Sigma_n \times \Sigma_n$, $(u, v) \xleftarrow{\$} \in \mathbb{F}_2^n \times \mathbb{F}_2^n$
$c_1 = h(\pi||Ru^\top + Qv^\top), c_2 = h(\pi(u)), c_3 = h(\pi(x+u))$
$d_1 = h(\sigma||Ru^\top + Qv^\top), d_2 = h(\sigma(v)), d_3 = h(\sigma(y+v))$
2. $\mathcal{P}$ sends $c_1, c_2, c_3, d_1, d_2, d_3$.
3. $\mathcal{V}$ sends $\mathsf{ch} \xleftarrow{\$} \{0, 1, 2\}$
4. 3 cases for $\mathcal{P}$:
$\mathsf{ch} = 0$. $\mathcal{P}$ sends $\{\pi, \sigma, u, v\}$
$\mathsf{ch} = 1$. $\mathcal{P}$ sends $\{\pi, \sigma, x+u, y+v\}$
$\mathsf{ch} = 2$. $\mathcal{P}$ sends $\{\pi(x), \pi(u), \sigma(y), \sigma(v)\}$

5. 3 cases for $\mathcal{V}$:
$\mathsf{ch} = 0$: $\mathcal{V}$ checks c_1, c_2, d_1, d_2
$\mathsf{ch} = 1$: $\mathcal{V}$ checks c_1, c_3, d_1, d_3
$\mathsf{ch} = 2$: $\mathcal{V}$ checks c_2, c_3, d_2, d_3
and $\omega t(\pi(x)) = \omega t(\sigma(y)) = \omega$
If all checks passed, $\mathcal{V}$ accepts $\mathcal{P}$'s knowledge.

6. Additional checks for $\mathcal{V}(y')$:
$\mathsf{ch} = 0$: checks if $d_3 = h(\sigma(y' + v))$
$\mathsf{ch} = 1$: checks if $d_2 = h(\sigma(y' + (y + v)))$
$\mathsf{ch} = 2$: no additional check
If additional check passed, $\mathcal{V}$ accepts y'.

Figure 5.2: Concatenated Stern's Protocol ($\mathcal{CSP}$)

5.3.2 Testable weak Zero Knowledge

Introduced in [55], ZK proofs allow a prover $\mathcal{P}$ to convince a verifier $\mathcal{V}$ of the veracity of a statement $\mathcal{S}$, without leaking any additional information on the proof but its length. Since its introduction, this kind of proofs has been widely applied to digital identification protocols, and by extension signature schemes using the Fiat-Shamir paradigm [47]. As recalled in Chapter 1, these proofs satisfy the following three properties:

- **Completeness.** If $\mathcal{S}$ is true, the honest verifier will be convinced except with negligible probability;
- **Soundness.** If $\mathcal{S}$ is false, no cheating prover can convince the honest verifier that it is true except with negligible probability;
- **Zero-Knowledge.** Anything that is feasibly computable from the proof is also feasibly computable from the assertion itself.

A classical variant of ZK proofs generally used is Witness-Indistinguishable (WI) proofs [54], where the ZK requirement is roughly weakened into "seeing the proof does not provide any information on the witness used". More formally, assuming a language $\mathcal{L}$ and witnesses z_0, z_1 satisfying the language, WI requires distributions D_0 and D_1 to be indistinguishable where $D_b = \{(z_0, z_1, \mathsf{Prove}(z_b, \mathcal{L}; \mu)), \text{for random string } \mu\}$, while not requiring proof simulatability anymore.

Nevertheless, our construction is going to need somewhat opposite requirements: we want to be able to build a simulator capable of simulating proofs, while allowing anyone possessing information to test if this piece of information is related to the witness used for generating the proof. Proofs of knowledge meeting these requirements will be called

Testable weak Zero-Knowledge (TwZK for short). The general idea behind TwZK is that there is a little information leaking compared to classical ZK: the fact that an attacker can test whether a particular information is related to the witness used in a proof. If this value is not known, an attacker should either test all possible values or find a particular one which, in both cases has to be hard. Hence an attacker who breaks the security of our model either breaks ROM security or is able to solve a computational problem.

As an example, considering Stern's protocol (Figure 5.1), this particular information about the witness is the witness itself. Indeed, as exhibited in Remark 1, if $\mathcal{V}$ knows the prover's secret z, he is able to trace this witness z by checking more requirements for cases $\mathsf{ch} = 0$ and $\mathsf{ch} = 1$.

We then propose the following definition.

Definition 10 (Testable weak Zero-Knowledge Proofs)**.** *A Testable weak Zero-Knowledge proof is defined through the following algorithms:*

- $\mathsf{TSetup}(1^\lambda)$*: generates the public parameters of the system among which some function* f *and possibly a simulation trapdoor* stk*;*
- $\mathsf{Prove}(z, \mathcal{L}; \mu)$*: generates a proof* Π *that a word* z *is in the language* $\mathcal{L}$ *using some random string* μ*;*
- $\mathsf{TVerif}(\Pi, \mathcal{L})$*: checks that the validity of a proof* Π *with respect to the language* $\mathcal{L}$*;*
- $\mathsf{TestWit}(y', \Pi, \mathcal{L})$*: for a valid proof* $\Pi \leftarrow \mathsf{Prove}(z, \mathcal{L}; \mu)$ *and* f *generated during* TSetup*, this algorithm checks, with negligible failure probability, if* $y' = f(z)$*;*
- $\mathsf{SProve}(\mathsf{stk}, \mathcal{L}; \mu)$*: generates a simulated proof of knowledge of a word in* $\mathcal{L}$ *using a trapdoor* stk*.*

In addition to classical correctness and soundness properties, we want weak indistinguishability of proof simulation: for a non empty language $\mathcal{L}$, a proof generated using a witness (case S_0) must be indistinguishable from one using the trapdoor (case S_1) meaning that $S_0 = \{\mathsf{Prove}(z, \mathcal{L}; \mu) | w \in \mathcal{L}$ and $S_1 = \{\mathsf{SProve}(\mathsf{stk}, \mathcal{L}; \mu)\}$ have to be indistinguishable.

$\mathrm{Exp}^{wS-b}_{\mathcal{P},\mathcal{A}}(\lambda)$
1. $\mathsf{params} \leftarrow \mathsf{TSetup}(1^\lambda)$
2. if $(b = 0)$, $\Pi^* \leftarrow \mathsf{Prove}(z, \mathcal{L}; \mu)$
3. else $\Pi^* \leftarrow \{\mathsf{SProve}(\mathsf{stk}, \mathcal{L}; \mu)\}$
4. $b' \leftarrow \mathcal{A}(\Pi^*, \mathsf{TestWit}(.))$
5. Return b'

Compared to what might be expected for Classical Zero-Knowledge, the previous property can only be achieved for some languages. Namely those where guessing the correct related information $f(z)$ is hard (either because there is no efficient way to search through the possible set, or simply because the set of potential $f(z)$ is hard to sample).

Original Stern's protocol is TwZK We consider here the particular case of the non randomized version of Stern's protocol to exhibit that it is TwZK instead of ZK. We do not formally describe here how Stern's original protocol fulfills Definition 10 since the goal is only to highlight that it satisfies the property of TwZK encompassed in weak indistinguishability of proof simulation described above. To cope with step 6 of protocol $\mathcal{CSP}$ (Figure 5.2), we propose the following additional checks for a verifier, with additional knowledge a vector z', to Stern's protocol (Figure 5.1):

- case $\mathsf{ch} = 0$. $\mathcal{V}(z')$ checks if commitment $c_3 = h(\pi(z' + u))$;
- case $\mathsf{ch} = 1$. $\mathcal{V}(z')$ checks if commitment $c_2 = h(\pi(z + u + z'))$;
- case $\mathsf{ch} = 2$. No additional test in this case.

In two out of three cases ($\mathsf{ch} = 0$ and $\mathsf{ch} = 1$), an additional check can be processed so that this step has a failure probability $\leqslant 1/3$. Hence, in the non-interactive case where several repetitions of the protocol are demanded, this failure probability can be made as small as wished to design an algorithm $\mathsf{TestWit}$. For a dishonest prover, there are 3 different cheating strategies depending on the value computed for c. We are going to focus on $c = 0$, the others are left to the sagacity of the reader. (It should be noted that for $c = 2$ no test is possible so the indistinguishability proof is even easier.)

We use the Random Oracle programmability on H to force the hash value to be 0. Following Stern's framework, in this case the SProve algorithm would have to pick an honest π and u, and a random vector z of weight ω (not satisfying the equation $Hz^\top = s$), and proceeds as expected. The difference in distributions between honestly generated proofs and simulated one comes from $h(\pi(z + u))$, which in the first case is correctly generated while in the other one is not. Assuming h is also a random oracle the value $h(\pi(z + u))$ is random compared to the rest of the view of the adversary.

Now the only way the adversary could possibly distinguish a value $h(\pi(z + u))$ computed for a z in the language from a random value, would be by querying $\pi(z+u)$ to the ROM. (Until now, the ROM programmability allows us to argue that the two visions are indistinguishable).

Using Random Oracle Observability, one can then monitor the calls to the random oracles made by the adversary, and for the π, u used in the challenge, parse his calls to define values z_is. In the eventuality the adversary queries the ROM on a z_i in the language (i.e. $Hz_i^\top = s$ and $\omega t(z_i) = \omega$), a simulator generating a random proof without knowing a word in the language can use the adversary's query to solve the Syndrome Decoding challenge.

Formal transform into TwZK proofs and security of protocol $\mathcal{CSP}$ are respectively studied in subsection 3.3 and 3.4. Similar results for Stern's non randomized protocol

could then easily be deduced.

5.3.3 From $\mathcal{CSP}$ to Testable weak Zero Knowledge Proofs

Even if some points may appear straightforward, we detail here how our protocol $\mathcal{CSP}$ (Figure 5.2) can be seen as TwZK proofs satisfying Definition 10.

Description of TSetup According to a security parameter λ, TSetup algorithm generates public parameters among which an $\mathcal{CSP}$-instance (H, ω, s) and some function f. The $\mathcal{CSP}$-instance is meant to be chosen difficult and defines the following language $\mathcal{L} = \{z = (x\|y), x, y \in \mathbb{F}_2^n : Hz^\top = s \wedge \omega t(x) = \omega t(y) = \omega\}$. The function f stipulates how much information $f(z)$ could be tested about z while knowing some side information (algorithm TestWit).

TSetup(1^λ)
1. $(H, s, \omega, f) \xleftarrow{\$} 1^\lambda$ where:
   - $H = (R\|Q)$ with R, $Q \in \mathcal{M}_{(n-k)\times n}(\mathbb{F}_2)$
   - $s \in \mathbb{F}_2^{n-k}$, $\omega \in \mathbb{N}$
   - $f : \mathbb{F}_2^{2n} \to \mathbb{F}_2^n$ such as $f(x\|y) := y$
2. Set $\mathcal{L} = \{z = (x\|y), x, y \in \mathbb{F}_2^n :$
   $Hz^\top = s \wedge [\omega t(x) = \omega t(y) = \omega]\}$.

Return $(H, s, \omega, f, \mathcal{L})$.

Figure 5.3: TSetup protocol

In the following, we denote by params the quantity $(H, s, \omega, f, \mathcal{L}) \leftarrow$ TSetup(1^λ).

Description of Prove and TVerif Algorithm Prove enables anyone that possesses an element of $\mathcal{L}$ to prove such a knowledge without revealing it. Nevertheless, contrary to ZK proofs, some controlled information can be leaked and later be tested through algorithm TestWit. Through Fiat-Shamir paradigm, algorithm Prove reaps benefit of the interactive protocol $\mathcal{CSP}$ to generate a proof of knowledge on an element of $\mathcal{L}$. It then processes as follows: it first pre-computes $6 \times l_\lambda$ commitments to be stored in cmt on which the random oracle is then applied. Finally, for each ternary symbol ch$[j]$, algorithm Prove sets the corresponding response rsp$[j]$, as specified by protocol $\mathcal{CSP}$, and outputs the proof (cmt, rsp) that will later be given to TVerif.

This latter unfolds the proof, computes ch $=$ H$_\lambda$(cmt) and checks consistency according to interactive protocol $\mathcal{CSP}$. We recall here that l_λ is the number of times protocol $\mathcal{CSP}$ should be repeated to ensure negligible error probability. Algorithms Prove and TVerif are depicted in Figure 5.4.

(a) $\mathsf{Prove}(z, \mathcal{L}; \mu)$
Parse $z = (x \| y)$.
1. Commitments Generation.
For $j = 1 \dots l_\lambda$
$(\pi, \sigma) \xleftarrow{\$} \Sigma_n \times \Sigma_n$, $(u, v) \xleftarrow{\$} \in \mathbb{F}_2^n \times \mathbb{F}_2^n$
$c_1 = h(\pi || Ru^\top + Qv^\top), c_2 = h(\pi(u))$,
$c_3 = h(\pi(x+u))$, $d_1 = h(\sigma || Ru^\top + Qv^\top)$,
$d_2 = h(\sigma(v)), d_3 = h(\sigma(y+v))$,
$\mathsf{cmt}[j] \leftarrow \{c_1, c_2, c_3, d_1, d_2, d_3\}$.
2. Challenge Oracle Generation.
$\mathsf{ch} := \mathsf{H}_\lambda(\mathsf{cmt}) \in \{0,1,2\}^{l_\lambda}$.
3. Response Generation.
For $j = 1 \dots l_\lambda$
3 cases:
$\mathsf{ch}[j] = 0$: $\mathsf{rsp}[j] = \{\pi, \sigma, u, v\}$
$\mathsf{ch}[j] = 1$: $\mathsf{rsp}[j] = \{\pi, \sigma, x+u, y+v\}$
$\mathsf{ch}[j] = 2$: $\mathsf{rsp}[j] = \{\pi(x), \pi(u), \sigma(y), \sigma(v)\}$
4. $\Pi \leftarrow (\mathsf{cmt}, \mathsf{rsp})$.
Return $\Pi = (\mathsf{cmt}, \mathsf{rsp})$.

(b) $\mathsf{TVerif}(\Pi, \mathcal{L})$:
1. Parse $\Pi = (\mathsf{cmt}, \mathsf{rsp})$.
2. Challenge Oracle Generation.
$\mathsf{ch} := \mathsf{H}_\lambda(\mathsf{cmt}) \in \{0,1,2\}^{l_\lambda}$
3. Commitments and Responses Checks.
For $j = 1 \dots l_\lambda$
Parse $\mathsf{cmt}[j] = \{c_1, c_2, c_3, d_1, d_2, d_3\}$.
3 cases according to $\mathsf{ch}[j]$:
$\mathsf{ch}[j] = 0$: $\mathcal{V}$ uses $\mathsf{rsp}[j]$ to check c_1, c_2, d_1, d_2
$\mathsf{ch}[j] = 1$: $\mathcal{V}$ uses $\mathsf{rsp}[j]$ to check c_1, c_3, d_1, d_3
$\mathsf{ch}[j] = 2$: $\mathcal{V}$ uses $\mathsf{rsp}[j]$ to check c_2, c_3, d_2, d_3
and $\omega t(\pi(x)) = \omega t(\sigma(y)) = \omega$
4. If all checks passed, Return 1.
Else Return 0.

Figure 5.4: Algorithms Prove and TVerif

Description of TestWit Contrary to classical ZK proofs, the newly introduced concept of Testable weak Zero Knowledge (Definition 10) states that some controlled information can leak. Indeed, algorithm TestWit will be able to decide if some side information (y' here) is related to the secret witness (z here), as stipulated by f, used to generate a valid proof $\Pi \leftarrow \mathsf{Prove}(z, \mathcal{L}; \mu)$. Similarly to the non randomized version of Stern's protocol (see Remark 5.2.2), it is possible to trace proofs outputted by Prove. Before depicting the algorithm TestWit, we first give the general idea. Let us consider a witness $z = (x \| y)$ used to generate a proof. Now, using f defined such as $f(z) = y$ (Figure 5.3), algorithm TestWit can check if $f(z) = y'$ without knowing z by running additional checks when unfolding a proof $(\mathsf{cmt}, \mathsf{rsp})$. Indeed, according to the value of $\mathsf{ch}[j]$, TestWit works as follows:

- $\mathsf{ch}[j] = 0$, parse $\mathsf{rsp}[j] = \{\pi, \sigma, u, v\}$ and check if $d_3 = h(\sigma(y' + v))$;
- $\mathsf{ch}[j] = 1$, parse $\mathsf{rsp}[j] = \{\pi, \sigma, x+u, y+v\}$ and check if $d_2 = h(\sigma(y' + y + v))$;
- $\mathsf{ch}[j] = 2$, no such additional check is possible in this case.

These additional checks pass if and only if $y = y'$ which means that some verifier $\mathcal{V}(y)$ can trace a user owning the secret witness $z = (x \| y)$. As in Stern's protocol, an additional check is possible in two out of three cases ($\mathsf{ch} = 0$ and $\mathsf{ch} = 1$) whereas no such test in proposed when $\mathsf{ch} = 2$. The failure probability of this additional checking step is then $\leqslant 1/3$. In the non-interactive case where several repetitions of the protocol are

demanded, this failure probability can be made as small as wished to design algorithm TestWit. We refer the reader to Figure 5.5 (a) for a description of TestWit.

Remark 2. *As it will be proven in next subsection, $\mathcal{CSP}$ is a prover verifier protocol with cheating probability of* $2/3$*. To provide convincing proofs, with negligible error probability, we then have to set* $l_\lambda = \lambda/log(3/2)$*. On the other hand, additional checks do not apply only when* $\mathsf{ch} = 2$ *so that proofs simulating* l_λ *iterations of* $\mathcal{CSP}$ *will also also lead to a negligible error probability in the case of TestWit.*

Description of SProve It enables anyone with the trapdoor simulation stk to program the random oracle to generate a proof without knowledge of the secret key z that appears indistinguishable to a fair one i.e. as if it was an output of Prove.

According to the SD-problem, it is difficult to find a vector z both of small weight 2ω and verifying $Hz^\top = s$. Nevertheless, it is easy to independently find a $z_1 = (x_1||y_1)$, with no weight constraint, such as ${Hz_1}^\top = s$ and a $z_2 = (x_2||y_2)$ such as $\omega t(x_2) = \omega t(y_2) = \omega$. Let us now consider a simulator not knowing a valid secret that can set the value of ch^* by programming the random oracle. Following algorithm Prove, the simulator randomly picks π, σ, u and v and accordingly sets the value cmt^*.

- $\mathsf{ch}^* = 0$: the simulator computes commitments with z_1 or z_2 playing the role of z and stores the following values in rsp^*: π, σ, u, v;
- $\mathsf{ch}^* = 1$: the simulator computes commitments with z_1 playing the role of z and stores the following values in rsp^*: π, σ, $x_1 + u$, $y_1 + v$;
- $\mathsf{ch}^* = 2$: the simulator computes commitments with z_2 playing the role of z and stores the following values in rsp^*: $\pi(x_2)$, $\pi(u)$, $\sigma(y_2)$, $\sigma(v)$.

When called by Algorithm TVerif, because of programmability the random oracle H_λ will output ch^* on input cmt^* so that the simulated proof Π^* set to $(\mathsf{cmt}^*, \mathsf{rsp}^*)$ clearly passes algorithm TVerif.

Algorithm SProve is formally described in Figure 5.5 (b).

5.3.4 Security of the Concatenated Stern's protocol

In this subsection we show that protocol $\mathcal{CSP}$ presented in Figure 5.2 is a TwZK protocol that ensures completeness, soundness and testable weak zero-knowledge. This protocol is a prover-verifier protocol with cheating probability equal to $2/3$ that needs to be repeated several times to decrease this cheating probability close to 0.

By design, it is straightforward that proving the security of protocol $\mathcal{CSP}$ enables to prove that algorithms defined in previous subsection fulfill Definition 10. Indeed,

(a) $\mathsf{TestWit}(y', \Pi, \mathcal{L})$
1. Check proof validity.
If $\mathsf{TVerif}(\Pi, \mathcal{L}) = 0$, return $\perp$.
2. Challenge Oracle Generation.
$\mathsf{ch} := \mathsf{H}_\lambda(\mathsf{cmt}) \in \{0,1,2\}^{l_\lambda}$
3. Additional Checks.
Parse $\Pi = (\mathsf{cmt}, \mathsf{rsp})$.
For $j = 1 \dots l_\lambda$
Parse $\mathsf{cmt}[j] = \{c_1, c_2, c_3, d_1, d_2, d_3\}$
3 cases according to $\mathsf{ch}[j]$:
$\mathsf{ch}[j] = 0$: parse $\mathsf{rsp}[j] = \{\pi, \sigma, u, v\}$
If $d_3 = h(\sigma(y' + v))$, return 1.
Else, return 0.
$\mathsf{ch}[j] = 1$: parse $\mathsf{rsp}[j] = \{\pi, \sigma, x + u, y + v\}$
If $d_2 = h(\sigma(y' + y + v))$, return 1.
Else, return 0.
$\mathsf{ch}[j] = 2$: does not apply
4. In any other case, return $\perp$.

(b) $\mathsf{SProve}(\mathsf{stk}, \mathcal{L}; \mu)$
Parse $\mathsf{params} = (H, s, \omega, f, \mathcal{L})$.
1. For $j = 1 \dots l_\lambda$
1.1. Fake secrets Generation.
$z_1 = (x_1 \| y_1) \xleftarrow{\$} \mathbb{F}_2^{2n}$ such as $Hz_1^\top = s$,
$z_2 = (x_2 \| y_2) \xleftarrow{\$} \mathbb{F}_2^{2n}$ such as $\omega t(x_2) = \omega t(y_2) = \omega$.
1.2. Simulated Proof Generation.
$(\pi, \sigma) \xleftarrow{\$} \Sigma_n \times \Sigma_n$, $(u, v) \xleftarrow{\$} \in \mathbb{F}_2^n \times \mathbb{F}_2^n$,
$\mathsf{ch}^*[j] \xleftarrow{\$} \{0,1,2\}$.
Following Step 1 and Step 3 of Figure 5.4 (a):
$\mathsf{ch}^*[j] = 0$: compute $\mathsf{cmt}^*[j]$ using z_1 or z_2
and set $rsp^*[j] = \{\pi, \sigma, u, v\}$
$\mathsf{ch}^*[j] = 1$: compute $\mathsf{cmt}^*[j]$ using z_1
and set $rsp^*[j] = \{\pi, \sigma, x_1 + u, y_1 + v\}$
$\mathsf{ch}^*[j] = 2$: compute $\mathsf{cmt}^*[j]$ using z_2 and
set $rsp^*[j] = \{\pi(x_2), \pi(u), \sigma(y_2), \sigma(v)\}$
2. Use stk to program the random oracle to have:
$\mathsf{H}_\lambda(\mathsf{cmt}^*) = \mathsf{ch}^* \in \{0,1,2\}^{l_\lambda}$.
Return $\Pi^* = (\mathsf{cmt}^*, \mathsf{rsp}^*)$.

Figure 5.5: Algorithms TestWit and SProve

completeness and soundness of protocol $\mathcal{CSP}$ imply that algorithms Prove, TVerif and TestWit are well defined while showing that proofs outputted by Prove and SProve are indistinguishable ensures TwZK of protocol $\mathcal{CSP}$. Let us notice that we extend methodology of subsection 3.2 about TwZK of non randomized Stern's protocol to prove the TwZK of protocol $\mathcal{CSP}$.

Theorem 4. *The Concatenated Stern's protocol $\mathcal{CSP}$ is a prover verifier Testable weak Zero-Knowledge protocol with cheating probability $2/3$ verifying properties of completeness, soundness and Testable weak Zero-Knowledge in the ROM and assuming the hardness of the SD-problem.*

Proof We prove Theorem 4 through 3 lemmas in which we respectively discuss completeness, soundness and Testable weak Zero-Knowledge.

Completeness To ensure the completeness of our scheme, we exhibit that for a prover and a verifier honestly proceeding $\mathcal{CSP}$, it always succeeds.

Lemma 1. *If $\mathcal{P}$ and $\mathcal{V}$ honestly execute $\mathcal{CSP}$, we have for any round that $Pr[\mathcal{CSP}_{\mathcal{P},\mathcal{V}} = Accept] = 1$.*

Proof. The proof of the completeness is straightforward. The only subtlety might be to notice that z is a valid secret, which means that $Rx^\top + Qy^\top = s$. Then, for checking c_1 and c_2 in the case $\mathsf{ch} = 1$, we have to use that $R(x+u)^\top + Q(y+v)^\top - s = Rx^\top + Ru^\top + Qy^\top + Qv^\top - s = Ru^\top + Qv^\top$. □

Soundness To ensure the soundness of our scheme, we ensure that a cheating prover $\mathcal{P}$ cannot convince $\mathcal{V}$ that he knows a valid secret z when he does not.

Lemma 2. *If a cheating prover $\mathcal{P}$ and an honest verifier $\mathcal{V}$ execute $\mathcal{CSP}$, then for any round, we have that: $Pr[\mathcal{CSP}_{\mathcal{P},\mathcal{V}} = Accept] = 2/3$.*

Proof. We will prove that $\mathcal{P}$ has a maximum probability probability $2/3$ to cheat. For $\mathcal{P}$ to cheat, he must be able to answer correctly to any challenge ch at each round whereas he does not know a valid z. To accept the protocol, $\mathcal{V}$ should be able to verify all the hash values at the end of the interaction.

- $\mathsf{ch} = 0$: $\mathcal{P}$ reveals two permutations π_1, σ_1 and two vectors u and v respectively simulating values of π, σ, u and v in a fair protocol (we do not make this precision in the following);
- $\mathsf{ch} = 1$: $\mathcal{P}$ reveals two permutations π_2, σ_2 and two vectors u' and v';
- $\mathsf{ch} = 2$: $\mathcal{P}$ reveals vectors X, U, Y, V with $\omega t(X) = \omega t(Y) = \omega$.

Those values must verify hash values involved in the protocol. Let us consider a knowledge extractor $\mathcal{E}$ rewinding the random oracle. Since h is a random oracle (one cannot find a collision on it), we have the following straightforward results: $\pi_1 = \pi_2$, $\sigma_1 = \sigma_2$ and $Ru^\top + Qv^\top = Ru'^\top + Qv'^\top - s$ which leads to $R(u' - u)^\top + Q(v' - v)^\top = s$.
By exploiting consistency between different forms of c_2 and d_2, $\mathcal{E}$ gets that: $U = \pi(u)$ and $V = \sigma(v)$. Thanks to d_3 and c_3, we get: $\pi(u') = X + U$ and $\sigma(v') = Y + v$. Finally, by setting $z^* = (u' - u \| v' - v)$, $\mathcal{P}$ enabled $\mathcal{E}$ to find z^* verifying $Hz^{*\top} = s$ and $\omega t(z^*) = 2\omega$ i.e. $\mathcal{E}$ solved the SD-problem.

Then, $\mathcal{P}$ cannot anticipate the 3 challenges and the protocol clearly outputs *Accept* with a maximum probability of $2/3$ (cases $\mathsf{ch} = 0$ and $\mathsf{ch} = 1$ or cases $\mathsf{ch} = 0$ and $\mathsf{ch} = 2$ according to the strategy of the cheating prover).

□

Testable weak Zero-Knowledge We will now prove the Testable weak Zero-Knowledge property of our scheme. The idea is to prove that a verifier cannot learn nothing more from a fair execution than prover's secret is correct and whether the auxiliary value he knows is related to prover's secret or not.

Lemma 3. *The Concatenated Stern's protocol $\mathcal{CSP}_{\mathcal{P},\mathcal{V}}$ depicted in Figure 5.2 is a Testable weak Zero-Knowledge proof in the random oracle model assuming the hardness of the SD-problem.*

Proof. Given a valid proof Π^* (i.e. for which TVerif outputs 1), there are two possibilities for an adversary $\mathcal{A}$: algorithm $\mathsf{TestWit}$ either outputs 0 or 1. In the first case, honestly generated proofs and simulated ones are clearly random since commitments stored in cmt are obtained via the random oracle h and values to be revealed, stored in rsp, are designed to appear random (the security of all Stern like protocols relies on this fact).

We now focus on the case where case $\mathsf{TestWit}$ outputs 1.

If Π^* was generated by algorithm SProve, there are three different cheating strategies:

- $\mathsf{ch} = 0$: the simulator runs SProve to output Π^*. While generating this proof (see Figure 5.5 (b)), the simulator implicitly defined the SD-instance (Q, s_1, ω) where $s_1 = Qy_1^\top$ for a certain y_1 with no constraint on the weight ($\omega t(y_1) \neq \omega$ a priori). The adversary will first check the proof by running TVerif and he additionally runs $\mathsf{TestWit}$ using its auxiliary knowledge y' to check the consistency of d_3. Thanks to RO observability, the simulator can monitor the calls made by the adversary to h. Now when $\mathcal{A}$ calls h to run $\mathsf{TestWit}$, the simulator can parse these calls to identify y'. Since y' is such as $\mathsf{TestWit}(y', \Pi^*, \mathcal{L}) = 1$, the simulator is able to find a vector y' of weight ω such as $s_1 = Qy'^\top$ and thus solving the SD-instance (Q, s_1, ω).

- $\mathsf{ch} = 1$: this time, the simulator implicitly defined the SD-instance (Q, s_2, ω) while generating Π^*, where $s_2 = Qy_2^\top$ for a certain y_2 with $Qy_2^\top \neq Qy^\top$ a priori. Once again, the RO observability enables the simulator to monitor the calls made by the adversary to h. Now when the adversary calls h to run $\mathsf{TestWit}$ to additionally check d_2, the simulator can parse these calls to identify y'. Since y' is such as $\mathsf{TestWit}(y', \Pi^*, \mathcal{L}) = 1$, the simulator is able to find a vector y' of weight ω such as $s_2 = Qy'^\top$ and thus solving the SD-instance (Q, s_2, ω).

- $\mathsf{ch} = 2$: the algorithm $\mathsf{TestWit}$ does not apply in this case and indistinguishability is straightforward.

Now, if this valid proof Π^* was generated by algorithm Prove, we consider the following game.

$\mathcal{G}$ The challenger supposed to run Prove will attribute random values to commitments stored in cmt except for the values of d_3 in the case $\mathsf{ch} = 0$ and d_2 in the case $\mathsf{ch} = 1$ for which he computes values according to $\mathcal{CSP}$.

Because of h modeled as a random oracle, to $\mathcal{A}'$s view, this game $\mathcal{G}$ is indistinguishable from an honest Prove procedure. Plus, for the same reason, it is also indistinguishable from the previous case with SProve (all commitments stored in cmt always appear random).

Hence, we can consider that $\mathcal{A}$ will never provide a word y' such as $\mathsf{TestWit}(y', \Pi^*, \mathcal{L}) = 1$ since it leads to breaking the SD-problem. Finally, under the ROM and the SD-problem, we can consider that distributions S_0 and S_1 are indistinguishable.

□

5.4 Definition and Security Model

In this section, we give the definition of the group signature we will rely on and the associated security model.

A group signature scheme [49] is a protocol which allows members of a group to individually issue signatures on behalf of the group in an anonymous but revocable way: an opener is able to revoke anonymity of the actual signer in case of abuse. Several steps have been made in the study of those protocols: Bellare et al. [163] first gave formal security properties of group signature. Later, Bellare, Shi and Zhang (BSZ model) [161]extended this model to dynamic groups, emphasizing the importance of unforgeability and anonymity. Numerous efficient group signatures schemes such as [164, 165, 166] using bilinear maps were proposed but only secure in a relaxation of these models. While recent post-quantum group signature schemes, namely lattice-based, such as [171, 172, 173, 174, 175] only satisfy the static model of [163], we propose a scheme that we will define as *weakly dynamic* with the classic properties of *anonymity*, *traceability* and *non-frameability*.

5.4.1 Definition

Let us precise that our scheme only involves three entities with a single authority: the group manager. Indeed, in our model, the group manager will both participate in issuing users' secret keys and revoking anonymity (see Section 5) without impacting security (see Section 6). Consequently, the algorithm *Judge* present in dynamic models does not appear in our model. Under the existence of a Public Key Infrastructure (PKI) for exchanges between users and authorities, we adapt the BSZ model to propose the following definition.

Definition 11. *A group signature scheme* $\mathcal{GS} = (\mathsf{GSetup}, \mathsf{Join}, \mathsf{GSign}, \mathsf{GVerif}, \mathsf{Open})$ *is a sequence of protocols such as:*

- $\mathsf{GSetup}(1^\lambda)$*: this algorithm generates public parameters of the system* $\mathsf{gparams}$*, the group public key* gpk *and the group manager secret key* $\mathsf{gmsk} = (\mathsf{trk}, \mathsf{skO})$ *made of a trapdoor key* trk *and the opening key* skO*;*

- $\mathsf{Join}(\mathcal{U}_i)$*: this is an interactive protocol between a user* $\mathcal{U}_i$ *and the group manager owning* trk*. At the end of the protocol, the user obtains a secret signing key* $\mathsf{sk}[i]$*. The group manager adds the new user* $\mathcal{U}_i$ *and updates* skO*;*

- $\mathsf{GSign}(\mathsf{gpk}, \mathsf{sk}[i], m; \mu)$*: to sign a message* m*, the user uses his secret key* $\mathsf{sk}[i]$ *and some randomness* μ *to output a signature* Σ *valid under the group public key* gpk*;*

- $\mathsf{GVerif}(\mathsf{gpk}, m, \Sigma)$*: anybody should be able to verify the validity of the signature* Σ *on the message* m *with respect to* gpk*. It thus outputs* 1 *if the signature is valid, and 0 otherwise;*

- $\mathsf{Open}(\mathsf{skO}, \mathsf{gpk}, m, \Sigma)$*: for a valid signature* Σ *with respect to* gpk*, the group manager can provide the signer identity : it thus outputs the user* $\mathcal{U}_i$ *when it succeeds and 0 otherwise.*

5.4.2 Security Model

We define our security notions in a game-based way defined in Figures 5.6 and 5.7. To be claimed secure, a group signature scheme has to prove its *correctness* and fulfill three properties: *anonymity*, *traceability* and *non-frameability*.

Correctness The *correctness* notion guarantees that honest users should be able to generate valid signatures, and the opener should then be able to revoke anonymity of the signers.

Unforgeability

Informally, the unforgeability notion guarantees that no one can produce a valid signature that cannot be opened in convincing way (traceability) and that no one can produce a signature on behalf of some group member (non-frameability).

In the following experiments, to join the group, an adversary runs the oracle $\mathcal{O}^{\text{joinP}}$ (passive join) to create an honest user for whom it does not know the secret key: the index i is added to the HU (Honest Users) list. For users whose secret keys are known to the adversary, we let the adversary play on their behalf. For honest users, the adversary can interact with them, granted some oracles:

- $\mathcal{O}^{\text{corrupt}}(i)$, if $i \in \mathsf{HU}$, provides the secret key $\mathsf{sk}[i]$ of this user. The adversary can now control it. The index i is then moved from HU to the list of corrupted users CU;

- $\mathcal{O}^{\text{sign}}(i, m)$, if $i \in \mathsf{HU}$, plays as the honest user $\mathcal{U}_i$ would do in the signature process. Then i is appended to the list $S[m]$;

- the oracle $\mathcal{O}^{\text{open}}$ which, on input (m, Σ) returns $\mathsf{Open}(skO, gpk, m, \Sigma)$.

(a) Experiment $Exp^{tr}_{\mathcal{GS},\mathcal{A}}(\lambda)$
1. $(\mathsf{gpk}, \mathsf{gmsk} = (\mathsf{trk}, \mathsf{skO})) \leftarrow \mathsf{GSetup}(1^\lambda)$
2. $(m, \Sigma) \leftarrow \mathcal{A}(\mathsf{gpk} : \mathcal{O}^{\text{joinP}}, \mathcal{O}^{\text{corrupt}}, \mathcal{O}^{\text{sign}}, \mathcal{O}^{\text{open}})$
3. If $\mathsf{GVerif}(\mathsf{gpk}, m, \Sigma) = 0$, return 0.
4. If $\exists j \notin \mathsf{CU} \cup S[m]$,
   $\mathsf{Open}(\mathsf{skO}, \mathsf{gpk}, m, \Sigma) = j$,
   Return 1.
5. Else return 0.

$$Adv^{tr}_{\mathcal{GS},\mathcal{A}}(\lambda) = Pr[Exp^{tr}_{\mathcal{GS},\mathcal{A}}(\lambda) = 1]$$

(b) Experiment $Exp^{nf}_{\mathcal{GS},\mathcal{A}}(\lambda)$
1. $(\mathsf{gpk}, \mathsf{gmsk} = (\mathsf{trk}, \mathsf{skO})) \leftarrow \mathsf{GSetup}(1^\lambda)$
2. $(m, \Sigma) \leftarrow \mathcal{A}(\mathsf{gpk}, \mathsf{skO} : \mathcal{O}^{\text{joinP}}, \mathcal{O}^{\text{corrupt}}, \mathcal{O}^{\text{sign}})$
3. If $\mathsf{GVerif}(gpk, m, \Sigma) = 0$, return 0.
4. If $\exists i \in \mathsf{HU} \setminus S[m]$,
   $\mathsf{Open}(\mathsf{skO}, \mathsf{gpk}, m, \Sigma) = i$,
   Return 1.
5. Else return 0.

$$Adv^{nf}_{\mathcal{GS},\mathcal{A}}(\lambda) = Pr[Exp^{nf}_{\mathcal{GS},\mathcal{A}}(\lambda) = 1]$$

Figure 5.6: Unforgeability Notions

Traceability and Non-Frameability

Traceability (Figure 5.6 (a)) says that nobody should be able to produce a valid signature that cannot be opened in a convincing way. Furthermore, non-frameability (Figure 5.6 (b)) guarantees that no dishonest player (even the authorities, i.e. the Group Manager GM) will be able to frame an honest user: an honest user that does not sign a message m should not be convincingly declared as a possible signer, non-frameability also shows that the group manager cannot cheat. We thus say that:

- $\mathcal{GS}$ is *traceable* if, for any polynomial adversary $\mathcal{A}$, the advantage $Adv^{tr}_{\mathcal{GS},\mathcal{A}}(\lambda)$ is negligible;
- $\mathcal{GS}$ is *non-frameable* if, for any polynomial adversary $\mathcal{A}$, the advantage $Adv^{nf}_{\mathcal{GS},\mathcal{A}}(\lambda)$ is negligible.

In both games, the adversary generates a signature Σ on a message m of its choice. In the latter game, the adversary itself can play the role of the opener, trying to frame an honest user i.

Anonymity

Given two of honest users i_0 and i_1, the adversary should not have any significant advantage in guessing which one of them have issued a valid signature.

The adversary can interact with honest users as before (with $\mathcal{O}^{\text{corrupt}}$ and $\mathcal{O}^{\text{sign}}$), but the challenge signature is generated using the interactive signature protocol GSign, where the adversary plays the role of the corrupted users, but honest users are activated to play their roles.

$\mathcal{GS}$ is *anonymous* if, for any polynomial adversary $\mathcal{A}$, the advantage $Adv^{anon}_{\mathcal{GS},\mathcal{A}}(\lambda)$ is negligible (Figure 5.7). We will see that our anonymity notion can be related to *selfless-anonymity* introduced in [165] where a user can check if he is the signer of a given signature and learns nothing about it else. The *full-anonymity* notion, required by the BSZ model,

Experiment $Exp^{anon-b}_{\mathcal{GS},\mathcal{A}}(\lambda)$
1. $(\mathsf{gpk}, \mathsf{gmsk} = (\mathsf{trk}, \mathsf{skO})) \leftarrow \mathsf{GSetup}(1^\lambda)$
2. $(m, i_0, i_1) \leftarrow \mathcal{A}(\mathsf{gpk} : \mathcal{O}^{\text{joinP}}, \mathcal{O}^{\text{corrupt}}, \mathcal{O}^{\text{sign}}, \mathcal{O}^{\text{open}})$
3. $\Sigma \leftarrow \mathsf{GSign}(gpk, i_b, m, sk[i])$
4. $b' \leftarrow \mathcal{A}(\Sigma : \mathcal{O}^{\text{joinP}}, \mathcal{O}^{\text{corrupt}}, \mathcal{O}^{\text{sign}}, \mathcal{O}^{\text{open}})$
5. If $i_0 \notin \mathsf{HU}$ or $i_1 \notin \mathsf{HU}$, return 0.
6. Return b'.

$$Adv^{anon}_{\mathcal{GS},\mathcal{A}}(\lambda) = Pr[Exp^{anon-1}_{\mathcal{GS},\mathcal{A}}(\lambda) = 1] - Pr[Exp^{anon-0}_{\mathcal{GS},\mathcal{A}}(\lambda) = 1]$$

Figure 5.7: Anonymity Notion

means that anonymity is guaranteed even if the adversary is granted access to all users' private keys (even the challenged one) and the oracle $\mathcal{O}^{\text{open}}$ (excepted on the challenge signature).

A *weakly* dynamic model The BSZ model [161] basically defines two ways for an adversary $\mathcal{A}$ to add new group members: a passive one where an honest user is added and might be corrupted later while a kind of active join protocol enables $\mathcal{A}$ to add an already corrupted member (this user is immediately added to CU), for which he knows its secret key. In this latter case, the BSZ model can still provide non-frameability. Giving such an active join to an adversary $\mathcal{A}$ in our case is equivalent to assuming that $\mathcal{A}$ knows the trapdoor key trk. Nevertheless, in Section 5.6, we ensure the security of our scheme in a scenario in which $\mathcal{A}$ should never access to trk. This explains why $\mathcal{A}$ is only given skO instead of gmsk in the non-frameability case. This difference between our model and the BSZ model led us to define our scheme as *weakly-dynamic*.

Definition 12. *A group signature scheme verifying security notions of Figures 5.6 and 5.7 is said to be securely weakly-dynamic.*

5.5 Our Code-Based Group Signature

We now present our group-signature scheme satisfying Definition 11. To fix ideas, we first present an high level overview and then, we describe more precisely the operation of the different algorithms.

5.5.1 High Level Overview

Actors Our scheme brings into play:

- a group manager (GM): the single authority of our scheme. It runs the GSetup algorithm, adds new members to the group (algorithm Join protocol) and opens signatures (algorithm Open);

- group members: also referred as users who can sign on behalf of the group (algorithm GSign);
- outsiders: they do not belong to the group but can verify a signature granted the group public key gpk (algorithm GVerif).

The main idea of our scheme consists in building an offset collision of two syndromes associated to two different matrices to instantiate $\mathcal{CSP}$. We recall here that the trapdoor matrix Q enables to invert syndromes so that a classic scenario could be as follows.

First Step GM generates Q a trapdoor matrix with trk the corresponding trapdoor key and a random syndrome s_G. $\mathcal{U}$ chooses a random vector x of weight ω and computes $s = Rx^\top$. $\mathcal{U}$ then sends s to GM who uses its trapdoor key to compute, via algorithm Inv, y such as: $Qy^\top = s + s_G$ and $\omega t(y) = \omega$. If this steps fails, $\mathcal{U}$ chooses a new random value for x until a valid antecedent y can be found.
Then, GM returns y to $\mathcal{U}$ who finally forms its secret signing key $z = (x\|y)$. It is important to notice that half of the secret key, namely x, is only known by $\mathcal{U}$ himself; it will ensure non-frameability of our scheme.

Second Step $\mathcal{U}$ owns a secret key $z = (x\|y)$ such as $\omega t(x) = \omega t(y) = \omega$ and $Hz^\top = Rx^\top + Qy^\top = s_G$. This situation fits within the model of the protocol $\mathcal{CSP}$. Every group member then owns its secret key which leads to the group common syndrome s_G, as depicted in Figure 5.5.1.

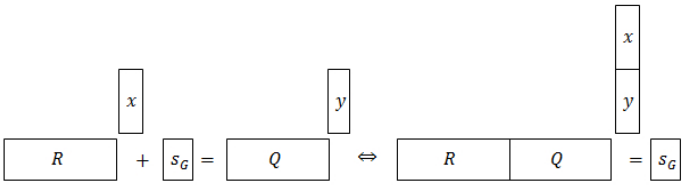

Figure 5.8: High level Overview

Third Step In case of doubt, GM should be able to revoke anonymity of group signatures. In the present case, GM knows y, the second part of the secret key of every user $\mathcal{U}$; this point will enable him to open signatures (algorithm *Open*).

5.5.2 Operation of our Scheme

In the CRS model, anyone is able to to generate $R = h'(\mathsf{crs})$.

We first describe algorithms GSetup, Join and summed them up in Figure 5.9. GSetup is performed by the group manager while Join($\mathcal{U}_i$) is an interactive protocol between a candidate $\mathcal{U}_i$ for joining the group and the group manager GM.

On adapting our TwZK protocol Our group signature scheme will mainly rely on the TwZK proofs described in Section 5.3. To cope with Fiat-Shamir paradigm in the context of a group signature scheme, we adapt algorithms Prove, TVerif and TestWit (Figures 5.4 and 5.5 (a)) so that step 2 of all these algorithms takes into account the message to sign m during the challenge generation. More precisely, algorithms Prove, TVerif and TestWit will take as additional input the message m so that ch is now equal to $\mathsf{ch} = \mathsf{H}_\lambda(m, \mathsf{cmt})$ instead of simply $\mathsf{H}_\lambda(\mathsf{cmt})$. We respectively denote these (slightly) modified algorithms $\mathsf{Prove}^{gs}(m, ., .; .)$, $\mathsf{TVerif}^{gs}(m, ., .)$ and $\mathsf{TestWit}^{gs}(m, ., .)$.

Description of GSetup Algorithm The GSetup algorithm is executed by the group manager GM taking as input a security parameter λ. It randomly generates a *trapdoor matrix* (according to Definition 9) Q and the corresponding trapdoor key trk. It also chooses a random syndrome $s_G \in \mathbb{F}_2^{n-k}$ that will constitute the group public identity and initializes $\mathsf{gmsk} = (\mathsf{trk}, \mathsf{skO})$ where skO will be its opening key. Finally, GM publishes global parameters $\mathsf{gparams} = (\lambda, n, k, \omega) \in \mathbb{N}^4$ and the group public key $\mathsf{gpk} = (H = (R\|Q), s_G, \omega)$ where R and $Q \in \mathcal{M}_{(n-k)\times n}(\mathbb{F}_2)$. This algorithm implicitly covers the run of a TSetup algorithm.

Description of Join Algorithm To proceed the *Join* protocol, GM and $\mathcal{U}_i$ behave as following: $\mathcal{U}_i$ randomly chooses a vector $x_i \in S_\omega^n$ and computes $s_i = Rx_i{}^\top$. Then, he sends s_i to GM who uses its trapdoor key trk to compute y_i verifying: $s_i + s_G = Qy_i{}^\top$ and $\omega t(y_i) = \omega$. GM responds y_i to $\mathcal{U}_i$. Finally, $\mathcal{U}_i$ forms $sk[i] = (x_i\|y_i)$ and GM updates $\mathsf{skO}[i] = y_i$. The reader should notice that this methodology may fail for two reasons. On the one hand, algorithm Inv can fail when computing antecedent y_i. On the other hand, when such an y_i is successfully computed, GM should first check that y_i was not already attributed to another user (GM ensures that all y_i's are different to further revoke anonymity). In both cases, GM tells $\mathcal{U}_i$ to choose another secret x_i.

In the end of this protocol, each user is given a secret key $z_i = (x_i\|y_i)$ of weight 2ω and verifying $Hz_i^\top = Rx_i^\top + Qy_i^\top = s_i + s_i + s_G = s_G$.

Description of GSign and GVerif Algorithms $\mathcal{CSP}$ (Figure 5.2) is an interactive TwZK protocol during which $\mathcal{P}$, which in fact consists, here, in the group G, proves to $\mathcal{V}$ the knowledge of a valid secret ensuring him that he belongs to the group associated to the common syndrome s_G.

(a) $\mathsf{GSetup}(1^\lambda)$

1. $\mathsf{gparams} = (\lambda, k, n, \omega) \in \mathbb{N}^4 \leftarrow 1^\lambda$
2. $(Q, \mathsf{trk}) \in \mathcal{M}_{(n-k)\times n}(\mathbb{F}_2) \times \mathcal{TRK} \leftarrow \mathsf{TrapGen}(1^\lambda)$
3. $s_G \xleftarrow{\$} \mathbb{F}_2^{n-k}$, $R = h'(crs)$
4. $\mathsf{gpk} \leftarrow (H = (R\|Q), s_G, \omega)$
5. $\mathsf{skO} \leftarrow \emptyset$
6. $\mathsf{gmsk} \leftarrow (\mathsf{trk}, \mathsf{skO})$
7. Output $(\mathsf{gparams}, \mathsf{gpk}, \mathsf{gmsk})$.

(b) $\mathsf{Join}(1^\lambda)$

	$user : \mathcal{U}_i$		GM
1.	$x_i \xleftarrow{\$} S_\omega^n$ $s_i = Rx_i^\top$	$\xrightarrow{s_i}$	
2.			Run $\mathsf{Inv}(Q, trk, s_i + s_G, \omega)$ If $\mathsf{Inv}(Q, trk, s_i + s_G, \omega) = \perp$, go to 1 If $\exists i_0 : \mathsf{skO}[i_0] = y_i$, go to 1 Else $\mathsf{skO}[i] = y_i$
3.	$sk[i] = (x_i\|y_i)$	$\xleftarrow{y_i}$	

Figure 5.9: *GSetup* and *Join* algorithms

As already mentioned, we use Fiat-Shamir paradigm to get a signature scheme. Signing then basically consists in outputting a proof of knowledge on a secret key that is linked to the message to sign m. Namely a group member $\mathcal{U}_i$ has to produce a signature Σ seen as a transcript $\Sigma = (\mathsf{cmt}, \mathsf{rsp})$ of the protocol $\mathcal{CSP}$ executed on public key gpk and small weight secret $\mathsf{sk}[i]$ for which each tuple $(\mathsf{cmt}[j], \mathsf{ch}[j], \mathsf{rsp}[j])$ simulates a fair execution of $\mathcal{CSP}$. It then suffices to run protocol Prove^{gs} on inputs m, $\mathsf{sk}[i]$ and $\mathcal{L}$.

For verification (algorithm GVerif), one has only to check a signature outputted by GSign which means checking the validity of some $\Sigma \leftarrow \mathsf{Prove}^{gs}(m, \mathsf{sk}[i], \mathcal{L}; \rho)$. We then apply algorithm TVerif^{gs} since it precisely aims at checking the validity of a proof outputted by Prove^{gs}.

Description of Open Algorithm We first recall that GM's opening key skO consists in the pool of y_is constituting half parts of users' secret keys made of $(x_i||y_i)$. According to Figure 5.5 (a), the knowledge of such values permits to run (potentially successfully) algorithm $\mathsf{TestWit}$ and then to decide if the tested value was the one used for generating the proof. The algorithm Open is then straightforward: given a message m and a signature $\Sigma = (\mathsf{cmt}, \mathsf{rsp})$, GM reads through the opening key skO until for some index i_0, $\mathsf{sk}[i_0] = y_{i_0}$ passes algorithm $\mathsf{TestWit}^{gs}$. In this case, GM is ensured that the actual signer is $\mathcal{U}_{i_0}$.

Remark 3. *By design of our scheme, any group member will then be able to open its own signatures which was referred to selfless-anonymity in [165].*

Algorithms GSign, GVerif and Open are depicted in Figure 5.10.

(a) $\mathsf{GSign}(\mathsf{gpk}, \mathsf{sk}[i], m, l_\lambda; \mu)$
1. Parse $\mathsf{gpk} = (H, s_G, \omega)$.
2. Set $\mathcal{L} = \{z = (x\|y), x, y \in \mathbb{F}_2^n :$
$Hz^\top = s_G \wedge \omega t(x) = \omega t(y) = \omega\}$
3. $\Sigma = (\mathsf{cmt}, \mathsf{rsp}) \leftarrow \mathsf{Prove}^{gs}(m, \mathsf{sk}[i], \mathcal{L}; \mu)$.
4. Return (m, Σ).

(b) $\mathsf{GVerif}(\mathsf{gpk}, m, \Sigma, l_\lambda)$
1. Parse $\mathsf{gpk} = (H, s_G, \omega)$.
2. Set $\mathcal{L} = \{z = (x\|y), x, y \in \mathbb{F}_2^n :$
$Hz^\top = s_G \wedge \omega t(x) = \omega t(y) = \omega\}$
3. $b \leftarrow \mathsf{TVerif}^{gs}(m, \Sigma, \mathcal{L})$
4. Return b.

(c) $\mathsf{Open}(\mathsf{skO}, \mathsf{gpk}, m, \Sigma)$
1. If $(GVerif(\mathsf{gpk}, m, \Sigma, l_\lambda) = 0)$, return 0.
2. Parse $\mathsf{gpk} = (H, s_G, \omega)$
3. Set $\mathcal{L} = \{z = (x\|y), x, y \in \mathbb{F}_2^n :$
$Hz^\top = s_G \wedge \omega t(x) = \omega t(y) = \omega\}$
4. For $i = 1 \ldots N$
$b \leftarrow \mathsf{TestWit}^{gs}(m, \mathsf{skO}[i], \Sigma, \mathcal{L})$
If $(b = 1)$, return $\mathcal{U}_i$.
5. If none $\mathsf{skO}[i]$ succeeds,
return 0.

Figure 5.10: GSign, GVerif and Open

5.5.3 Additional Properties

In this subsection, we briefly explain how the design of our group signature scheme might enable us to turn it into a traceable signature following the definition of [169] and how to handle revocation.

A traceable signature Extending group signatures schemes, Kiayias et al. suggested traceable signatures schemes in [169]. Such constructions enable the opening authority to delegate its opening or *tracing* capability to sub-openers so that they can trace suspicious users without letting them trace others. Usually, such a scheme provides two additional protocols compared to a classic group signature scheme: Trace and Claim.

Due to its construction, our scheme can easily be extended into traceable signature. Indeed if GM wants a sub-opener *So* to trace or look after $\mathcal{U}_{i_0}$ with secret key $z = (x_{i_0}||y_{i_0})$, he just needs to reveal him y_{i_0}. Now, whenever *So* is given a signature Σ, he uses y_{i_0} to apply algorithm Trace on Σ: it outputs 1 if and only if the issuer was indeed $\mathcal{U}_{i_0}$. In fact, he has to apply the methodology of algorithm Open but since he does not know skO, he cannot look over all users' tracing keys, then he will only be able to open the signature if it was issued by $\mathcal{U}_{i_0}$.

Furthermore, as pointed when defining anonymity, our scheme provides selfless-anonymity since any user will be able to prove that he is the actual signer of his own signature: this is exactly what algorithm Claim is meant to do.

Membership Revocation A crucial requirement for group signature schemes should be membership revocation. Indeed, once a group has been set up, one must be able to keep on trusting a group even in presence of misbehaving users; else, it means a new group should be regenerated to exclude them. When it comes to post-quantum constructions, only few lattice-based schemes [173, 175] handle this property by proceeding with verifier

local revocation (VLR). VLR requires the verifiers to possess some up-to-date revocation information, but not the signers. We follow the same methodology to provide membership revocation. As for the case of traceable signature, this revocation information, in our case, simply consists in the revoked user's tracing keys y_is.

We neither provide formal definitions nor detailed proofs here but the conversion of our scheme into in a traceable signature with membership revocation is rather straightforward. In particular, we refer the reader to the non-frameability property of our scheme to ensure that information released to potentials sub-openers or in the revocation list does not impact the security of our scheme.

5.6 Formal Security Analysis

In this section, we study the three requirements of anonymity, traceability and non-frameability previously defined (Figures 5.6 and 5.7) in order to claim secure our scheme (Definition 12). In the following, we consider $\mathcal{A}$ an adversary to our scheme and $\mathcal{B}$, an adversary to a difficult problem using advantage of $\mathcal{A}$'s possibilities.

Recalls on our group signature A signature consists in a transcript representing several repetitions of our Concatenated Stern's protocol (Figure 5.2). Then a signature Σ is a couple $(\mathsf{cmt}, \mathsf{rsp})$ such as:

$$(\mathsf{cmt}, \mathsf{rsp}) = (\mathsf{cmt}[1], \ldots, \mathsf{cmt}[l_\lambda], \mathsf{rsp}[1], \ldots, \mathsf{rsp}[l_\lambda]) \tag{5.1}$$

- $\mathsf{cmt}[j]$ consists in commitments of our Concatenated Stern's protocol (Figure 5.2) generated at j-th iteration;
- $\mathsf{ch}[j]$ the j-th symbol of $\mathsf{ch} = \mathsf{H}_\lambda(m, \mathsf{cmt}) \in \{0,1,2\}^{l_\lambda}$. It plays the role of the random challenge sent by the verifier in the interactive version of $\mathcal{CSP}$ at j-th iteration;
- $\mathsf{rsp}[j]$ is the answer to the challenge $\mathsf{ch}[j] \in \{0,1,2\}$.

How to generate a simulated signature The methodology for a simulator, without knowledge of a secret key, to generate a signature Π^* that appears valid (i.e. that passes algorithm GVerif) is the one used in SProve. For each iteration $j = 1 \ldots l_\lambda$, the simulator chooses a value for $\mathsf{ch}[j]$ and stores commitments and responses respectively in cmt and rsp according to methodology described in protocol SProve. Finally, the simulator programs the RO to set $\mathsf{H}_\lambda(m, \mathsf{cmt})$ to ch and outputs $\Sigma^* = (\mathsf{cmt}, \mathsf{rsp})$.

Hence, such a simulator will then produce a transcript looking fair to any verifier, without knowing any valid secret z.

5.6.1 Anonymity

We begin with the anonymity property.

Theorem 5. *If there exists an adversary $\mathcal{A}$ that can break the anonymity property of the scheme (Figure 5.7), then there exists an adversary $\mathcal{B}$ that can break the Testable weak Zero-Knowledge (TwZK) property of our Concatenated Stern's protocol $\mathcal{CSP}$.*

Proof. Through a sequence of games, we will exhibit that an adversary against our anonymity property would be able to break the TwZK property of our scheme.

$\mathcal{G}_0$ $\mathcal{B}$ first runs $\mathsf{GSetup}(\lambda)$. He gives gpk to $\mathcal{A}$ who has also access to oracles $\mathcal{O}^{\text{joinP}}$,$\mathcal{O}^{\text{corrupt}}$, $\mathcal{O}^{\text{sign}}$. Now $\mathcal{B}$ and $\mathcal{A}$ act as described in $Exp^{anon-0}_{\mathcal{GS},\mathcal{A}}(\lambda)$ (Figure 5.7). In this game, $\mathcal{B}$ will honestly challenge the adversary $\mathcal{A}$ on $b = 0$. At some point, $\mathcal{A}$ produces (m, i_0, i_1). $\mathcal{B}$ will then behave honestly by signing m outputting $\Sigma_0 = \mathsf{GSign}(\mathsf{gpk}, \mathsf{sk}[i_0], m; \mu)$.

$\mathcal{G}_1$ In this game, $\mathcal{A}$ and $\mathcal{B}$ behave the same way than in $\mathcal{G}_0$ with same knowledge and oracles provided to $\mathcal{A}$ with the exception of queries about $\mathcal{U}_{i_0}$. Indeed, when $\mathcal{A}$ produces (m, i_0, i_1), instead of generating Σ_0, $\mathcal{B}$ generates a simulated signature Σ^* by programming the random oracle H_λ accordingly.

$\mathcal{G}_2$ This game is the version of game $\mathcal{G}_0$ in which $\mathcal{B}$ challenges $\mathcal{A}$ on challenge $b = 1$; the rest consists in $Exp^{anon-1}_{\mathcal{GS},\mathcal{A}}(\lambda)$ (Figure 5.7). So, when $\mathcal{A}$ produces (m, i_0, i_1), $\mathcal{B}$ responds $\Sigma_1 = \mathsf{GSign}(\mathsf{gpk}, \mathsf{sk}[i_1], m; \mu)$.

Since the Concatenated Stern's protocol $\mathcal{CSP}$ is TwZK, we have, on the one hand, Σ_0 and Σ^* are statistically close to each other. On the other hand, for the exact same reason, Σ^* and Σ_1 are statistically close then Σ_1 is statistically close to Σ_0. Finally, we can deduce that $Exp^{anon-0}_{\mathcal{GS},\mathcal{A}}(\lambda)$ and $Exp^{anon-1}_{\mathcal{GS},\mathcal{A}}(\lambda)$ are indistinguishable which leads to that $Adv^{anon}_{\mathcal{GS},\mathcal{A}}(\lambda) = Exp^{anon-0}_{\mathcal{GS},\mathcal{A}}(\lambda) - Exp^{anon-1}_{\mathcal{GS},\mathcal{A}}(\lambda)$ is negligible. This terminates the proof. □

5.6.2 Soundness

The soundness analysis consists in proving traceability and non-frameability.
Through a methodology similar to [176], we will first begin to show how forging a signature could lead to breaking a computational problem.

Forging a signature

Let us suppose that $\mathcal{A}$ can forge a signature on user $\mathcal{U}_{i_0}$ supposed to be uncorrupted. It follows that $\mathcal{A}$ can produce $\Sigma = (\mathsf{cmt}, \mathsf{rsp})$ such as $\mathsf{GVerif}(\mathsf{gpk}, m, \Sigma) =$

1 without knowing $\mathsf{sk}[i_0]$. As recalled in (1), Σ is the following transcript $(\mathsf{cmt}[1], \ldots, \mathsf{cmt}[l_\lambda], \mathsf{rsp}[1], \ldots, \mathsf{rsp}[l_\lambda])$ simulating l_λ fair iterations of protocol $\mathcal{CSP}$.

If $\mathcal{A}$ can produce such a forgery, then $\mathcal{A}$ must have been able to successfully run l_λ iterations of $\mathcal{CSP}$ without knowing a valid secret whereas the cheating probability of $\mathcal{CSP}$ is 2/3 (Lemma 2). Then $\mathcal{A}$ has either broken the soundness of $\mathcal{CSP}$ or enabled the design of a knowledge extractor reaping benefits of the forgery to produce a valid solution z of the related SD instance.

Soundness Since traceability and non-frameability are two notions closely related, we treat them simultaneously. Indeed, both notions require the adversary $\mathcal{A}$ to produce a valid forgery Σ verifying $\mathsf{GVerif}(\mathsf{gpk}, m, \Sigma) = 1$. Nevertheless, breaking traceability implies for $\mathcal{A}$ to produce Σ such as the group manager could not trace it back to any group member whereas non-frameability requires to produce a signature that does trace back to an actual group member.

More precisely, if we consider that $\mathcal{A}$ attacks an honest user $\mathcal{U}_{i_0}$: to attack traceability, $\mathcal{A}$ should produce a forgery Σ on m such as:

$$\mathsf{GVerif}(\mathsf{gpk}, m, \Sigma) = 1 \wedge \mathsf{Open}(\mathsf{skO}, \mathsf{gpk}, m, \Sigma) = 0, \tag{5.2}$$

whereas to attack non-frameability its forgery Σ should verify:

$$\mathsf{GVerif}(\mathsf{gpk}, m, \Sigma) = 1 \wedge \mathsf{Open}(\mathsf{skO}, \mathsf{gpk}, m, \Sigma) = i_0, \tag{5.3}$$

with the obvious constraint in both cases that $\Sigma \neq \mathsf{GSign}(\mathsf{gpk}, \mathsf{sk}[i_0], m; \mu)$.

We now prove the traceability and the non-frameability of the proposed group signature scheme.

Theorem 6. *If there exists an adversary $\mathcal{A}$ against the traceability (Figure 5.6 (a)) (resp. the non-frameability, Figure 5.6 (b)) of the scheme, then we can build an adversary $\mathcal{B}$ that can either break the security of the Concatenated Stern's protocol $\mathcal{CSP}$ or the OMSD (resp. SD) problem.*

Proof. Let $\mathcal{A}$ be a PPT adversary attacking the traceability (resp. non-frameability) property of our scheme with advantage $\epsilon(\lambda)$. Through a sequence of games, we will show that if $\mathcal{A}$ is efficient, then it is possible for a simulator $\mathcal{B}$ to solve a difficult problem with non negligible probability related to ϵ.

$\mathcal{G}_0$ Following our group signature scheme, a simulator $\mathcal{B}$ runs algorithm $\mathsf{GSetup}(\lambda)$. He then chooses a user $\mathcal{U}_{i_0}$ on which $\mathcal{A}$ will be challenged. The following consists in the traceability (resp. non-frameability) game defined in Figure 5.6 (a) (resp. Figure 5.6

(b)); $\mathcal{B}$ then provides $\mathsf{gpk} = ((R||Q), \omega)$ (resp. $\mathsf{gpk} = ((R||Q), \omega)$ and skO) to $\mathcal{A}$, which has also access to oracles $\mathcal{O}^{\text{joinP}}$, $\mathcal{O}^{\text{sign}}$, $\mathcal{O}^{\text{open}}$ and $\mathcal{O}^{\text{corrupt}}$ (*resp.* $\mathcal{O}^{\text{joinP}}$, $\mathcal{O}^{\text{sign}}$ and $\mathcal{O}^{\text{corrupt}}$ since in this case, $\mathcal{A}$ knows skO and has no need for oracle *open*). For any query of $\mathcal{A}$, $\mathcal{B}$ responds honestly but the game aborts if $\mathcal{A}$ tries to corrupt $\mathcal{U}_{i_0}$.

At some point, $\mathcal{A}$ produces a forgery (m, Σ) under the condition that for all $i \in \mathsf{CU}$, signatures on m were never queried. As supposed earlier, the probability for $\mathcal{A}$ to have $\mathsf{GVerif}(\mathsf{gpk}, m, \Sigma)$ outputting 1 is ϵ.

$\mathcal{G}_1$ In this game, $\mathcal{B}$ still runs algorithm $\mathsf{GSetup}(\lambda)$ and chooses a user $\mathcal{U}_{i_0}$. $\mathcal{A}$ still knows gpk (resp. gpk and skO) with the same respective oracle accesses. The only difference from previous game is that whenever $\mathcal{A}$ queries oracle *sign* on user $\mathcal{U}_{i_0}$, $\mathcal{B}$ generates a simulated valid signature. Like previously, the game aborts $\mathcal{A}$ tries to corrupt $\mathcal{U}_{i_0}$.

At some point, $\mathcal{A}$ produces a forgery (m, Σ) under the condition that for all $i \in \mathsf{CU}$, signatures on m were never queried. Similarly to anonymity game (see subsection 5.6.1), signatures honestly produced on behalf of $\mathcal{U}_{i_0}$ in game $\mathcal{G}_0$ are indistinguishable from random ones produced in this game.

Hence, to $\mathcal{A}$'s view, games $\mathcal{G}_0$ and $\mathcal{G}_1$ are indistinguishable and we get that the probability for $\mathcal{A}$ to have $\mathsf{GVerif}(\mathsf{gpk}, m, \Sigma) = 1$ is still ϵ.

Under the soundness of $\mathcal{CSP}$, we now treat separately traceability (game $\mathcal{G}_1^{trac}$) and non-frameability (game $\mathcal{G}_1^{nf}$).

$\mathcal{G}_1^{trac}$ $\mathcal{A}$ has been given a forgery Σ verifying (2). Under the soundness of $\mathcal{CSP}$, $\mathcal{B}$ could thus apply the knowledge extractor algorithm $\mathcal{E}$ on Σ to generate a vector z^* passing protocol $\mathcal{CSP}$. All through this game, $\mathcal{A}$ may have queried for oracle *corrupt* on all users except $\mathcal{U}_{i_0}$ meaning that he may have obtained many secret keys solving the instance $(H, s_G, 2\omega)$. In other words, $\mathcal{A}$ has obtained the following OMSD instance $(H, s_G, 2\omega, \{\mathsf{usk}[i]\}_{i \in \mathsf{CU}})$. If $z^* \notin \{\mathsf{usk}[i]\}_{i \in \mathsf{CU}}$, $\mathcal{A}$ has then enabled $\mathcal{B}$ to find one more solution to the previous OMSD instance with probability directly related to ϵ. Else, $\mathcal{A}$ replays the game until $\mathcal{B}$ gets a new solution to the aforesaid OMSD instance.

$\mathcal{G}_1^{nf}$ In the case of non-frameability, $\mathcal{A}$ has also been given a forgery Σ verifying (3). This time, the knowledge of skO provides more information to $\mathcal{A}$ about users' secret keys. Indeed skO consists in all tracing keys (y_is), that is, half of every user's secret key. Writing $\mathsf{usk}[i] = (x_i||y_i)$ for all i, we have that $\mathcal{A}$ can compute $s_i = Qy_i^\top (= Rx_i^\top)$. Through oracle *corrupt*, $\mathcal{A}$ can learn the entire z_i for every user (different from $\mathcal{U}_{i_0}$) he might corrupt. We recall here that every

s_i, originally computed during the join procedure by $Rx_i^\top = s_i$ appears random since every x_i is randomly chosen in $\mathcal{S}_\omega^n$. In fact, $\mathcal{A}$ has obtained the following unsolved SD-instances $(R, s_i, \omega)_{i \in \mathsf{HU}}$ containing the particular one (R, s_{i_0}, ω).

Now, under the soundness of $\mathcal{CSP}$, $\mathcal{B}$, by programming the ROM, exploits Σ to get a vector $z^* = (x^*||y^*)$ from which he can issues signatures verifying (3) just like Σ. In particular, it means that applying algorithm Open on signatures issued with z^* returns i_0. It leads to $y^* = y_{i_0}$ and then that x^* is a solution to the SD instance (R, s_{i_0}, ω).

At the end of game $\mathcal{G}_1$, $\mathcal{A}$ has either broken the soundness of $\mathcal{CSP}$ or been able to solve a computational problem with non negligible probability related to ϵ. This terminates the proof.

□

5.7 Instantiation with the CFS scheme

Our scheme is generic and can be used with any trapdoor matrix. For coding theory based on Hamming metric, a possible trapdoor function is the CFS signature algorithm [58].

5.7.1 CFS Distinguishability and Security

The main idea of the CFS signature is to hide a Goppa code matrix with parameters $[2^m, 2^m - m\omega, 2\omega + 1]$ correcting up to ω errors. Parameters are chosen such that the probability to invert a syndrome is in $1/\omega!$; in practice, ω is of order $O(log(2^m)) = O(m)$. From the security discussion, the parameters have to be chosen so that the Syndrome Decoding problem for the matrix $H = (R\|Q)$ is difficult for a weight 2ω.

A recent result has proven that the public matrix of the CFS scheme was distinguishable [16] from a random matrix. Nevertheless, this result did not give rise to any attack on the scheme which then remains usable. Indeed, in practice and despite the aforesaid distinguisher, best attacks to the CFS problem are generic and treat the CFS matrix as a random one.

CFS-based problems and augmented-CFS matrix To be claimed secure (Section 5.6), our generic scheme does not require the public matrix to be indistinguishable from random but only to be secure against some computational problems defined in Section 5.2 (namely SD and OMSD problems). In the case of random codes, these problems are either proven hard (SD problem) or assumed to be (OMSD problem).

Assuming the putative security of the CFS scheme, we also consider that these problems are hard when instantiated with a CFS public matrix which can then be seen as a trapdoor matrix satisfying Definition 9.

This leads us to deal with a global public matrix of the form $\tilde{H} = (R\|Q)$ where R is random and Q is CFS public matrix; we then define *augmented-CFS* matrices as follows:

Definition 13. *Let $Q \in \mathcal{M}_{(n-k)\times n}(\mathbb{F}_2)$ a CFS public matrix and $R \in \mathcal{M}_{(n-k)\times n}(\mathbb{F}_2)$ a random binary matrix. We say that the following matrix $\tilde{H} = (R\|Q)$ is an augmented-CFS matrix.*

Since computational problems defined in Section 5.2 are also assumed to be hard using the CFS scheme, we can naturally extend this hardness to *augmented-CFS* matrices.

Remark 4. *K. P. Mathew et al. recently proposed a provably secure code-based signature scheme [180]. Their work is based on masking the public matrix of CFS scheme so that there may not exist any distinguisher with random matrices. Nevertheless, this leads to greater key sizes and since the classical CFS scheme is sufficient in our case, we focus on the latter when studying parameters.*

5.7.2 Parameters

We then instantiate our generic scheme with *augmented-CFS* matrices and consider, relying on previous subsection, the choice of parameters as if we were dealing with random ones. From the best known attacks [181, 63] we can choose $m = 20$ and $\omega = 10$. It leads to a matrix $\tilde{H}$ of length 2^{21} and dimension 200, with a security of 80 bits. The level of security can be improved over 80 bits by taking parameters with $m = 20$ and $\omega = 11$. The fact that the searched small weight vectors have a particular form since they have to be of same weight ω on the two parts of the matrix only decreases the complexity of the attacks at the margin. Indeed it decreases the number of possible solutions to the Syndrome Decoding problem only by a small factor, since random solutions of weight 2ω to the problem have a good probability to be of equal weight on each part of the matrix.

The number of syndromes $2^{m\omega}$, for a $[2^m, 2^m - m\omega, 2\omega+1]$ Goppa Code, gives an upper bound on the number of users. In fact, this number of users corresponds to the number of decodable random syndromes which is equivalent to $2^{m\omega}/\omega!$. In practice, one considers $\omega = O(m)$; hence, from the Stirling formula, it gives $2^{O(m^2)}$ possible users. Since public keys matrices and signatures sizes are proportional to the length of code 2^m, this leads sizes of order $N^{1/\sqrt{log(N)}}$, for N the number of group members.

Finally, parameters as chosen above lead to a signature (a transcript of proof of knowledge of a small word associated to $\tilde{H}$) of length roughly 20 megabytes and a public key of size 2.5 megabytes.

Notice that what takes time in the protocol is the computation of the CFS signature by the signer, but this is done only once for each member of the group when he enters the groups, and hence it is less important that this signature takes a little more time than usual signatures. At last the CFS signature scheme cannot find a preimage for any syndrome s, it does it only with probability $1/\omega!$, this fact can be managed through the sending of $\omega!$ different syndromes s in the Join process so that, on average, a preimage y by the CFS public matrix Q is found with a small failure probability, in which case the set-up process is started over. Since the syndrome s is computed randomly, it does not affect the security of the scheme.

As a final point, let us notice that in the quantum setting, it would be required to consider a security parameter λ greater than 80. Even if theoretically feasible, this may to lead extreme sizes of parameters in the case of CFS.

5.8 Conclusion

Basé sur la théorie des codes, notre signature de groupe est le premier schéma à la fois résistant à l'ordinateur quantique et dynamique[1].

L'idée de notre protocole est de construire une collision de deux syndromes associés à deux matrices différentes : une matrice aléatoire qui permet de générer un syndrome aléatoire à partir d'un mot de petit poids choisi par l'utilisateur et une matrice *à trappe* qui permettra de calculer un antécédent de petit poids dudit syndrome auquel une valeur fixe aura été ajoutée. La concaténation de ces deux mots de petit poids formera la clé de signature d'un utilisateur qui, via notre adaptation du protcole de Stern (*Concatenated Stern's Protocol*, Figure 5.2), lui permettra de convaincre un vérfieur que son secret est bien formé.

La sécurité de notre schéma repose un nouveau problème basé sur la théorie des codes défini comme étant le *One More Syndrome Decoding* et sur un nouveau type de preuve de connaissance, *Testable weak Zero-Knowledge*, pour lequel un vérifieur sera seulement capable d'apprendre si un témoin donné est associé ou non à celui du prouveur.

Malgré des tailles de clés importantes inhérentes à la métrique de Hamming (Section 5.7), notre schéma bénéficie de performances asymptotiques plus que satisfaisantes puisque tailles de clés et signatures seront proportionnelles à $N^{1/\sqrt{log(N)}}$. Ceci dit, et en dépit de son élégance et des nombreuses propriétés qu'il satisfait (*non-frameability*, dynamicité, traçabilité et révocabilité), l'instanciation proposée répond théoriquement à la limitation L9 mais ne permet pas d'envisager un déploiement industriel à court terme. Considérant L9 comme partiellement résolue, nous nous sommes

[1]Libert *et al.* ont depuis proposé un tel schéma basé sur les réseaux euclidiens [182]

alors penchés sur le cas de la métrique rang, également post-quantique, pour laquelle de récents cryptosystèmes prometteurs proposent des tailles de clés significativement moindres [20, 183, 184, 185].

Chapitre 6 :

Stade TA4 - Signature de groupe en métrique Rang

Contents

Comme son nom l'indique, la théorie des codes basés sur la métrique Rang diffère des codes correcteurs d'erreurs basées sur la métrique de Hamming par la métrique considérée. Le principal avantage de la cryptographie basée sur la métrique Rang réside dans la taille de clés réduites des cryptosystèmes associés [20, 183, 184, 185]. Ainsi, faisant suite à la conclusion du précédent chapitre, nous proposons ici le premier schéma de signature de groupe basé sur la métrique Rang. Dynamique dans une relaxation du modèle BSZ, il propose des tailles de clés et de signatures logarithmiques en le nombre des utilisateurs et dont les instanciations surpassent les constructions basées sur les réseaux et s'approchent des couplages bilinéaires [166]. À titre d'exemple, pour un niveau de sécurité de 100 bits, nous obtenons une instanciation qui menant à des signatures et une clé publique respectivement de tailles 550 kB et 5 kB.

Un tel résultat sera obtenu en adaptant le protocole $\mathcal{CSP}$ introduit au Chapitre 5 au cas de la métrique Rang, protocole que le paradigme de Fiat-Shamir permettra alors de convertir en une signature de groupe. De manière analogue, nous discuterons alors

la sécurité de notre schéma en proposant une adaptation de deux problèmes basés sur la métrique de Hamming que sont le problème OMSD (Chapitre 5) et le problème de décision du décodage par syndrome ou D-SD [186] (pour *Decision Syndrome Decoding*). La suite de ce chapitre, rédigée en anglais, a été présentée à la conférence *International Workshop on the Arithmetic of Finite Fields* (WAIFI 2016) et est le résultat de travaux réalisés conjointement avec Olivier Blazy, Stéphane Cauchie et Philippe Gaborit.

6.1 Introduction

Some new quantum resistant group signatures had been proposed between the end of Chapter 5 and Chapter 6. Mainly lattice-based [177, 187, 182], they come with new properties (message-dependent opening [177], no trapdoor matrix [187]) and achieve logarithmic complexities in the number of users [177, 182] while only one allows dynamic enrollment of new users [182]. In the end of this Chapter, Figure 6.7 then compares our work to its concurrent instantiations.

Our contributions.

Our rank-based construction constitutes the first post quantum group signature scheme to both enable enrollment of new users and enjoy practical parameters. Indeed, our scheme benefits from public keys sizes logarithmic in the number of group members, leading to an instantiation with signatures and public key, respectively of sizes 550 kB and 5 kB for a 100 bits security level. For such a purpose, we propose a novel approach while designing a (rank-based) Stern-like authentication protocol, referred as the Rank Concatenated Stern's Protocol. The key idea is to enable a verifier to check the weights of both parts of a split secret. This protocol is then turned into a group signature via Fiat-Shamir (FS) paradigm [47] to constitute a new tool in the growth of rank-based cryptography. We describe a generic scheme that we instantiate with the LRPC cryptosystem and RankSign scheme so that the practical security of our scheme relies on these aforesaid schemes and the two rank-based problems introduced along with this work, referred as the *One More Rank Syndrome Decoding* (OMSD) and the *Decision Rank Syndrome Decoding* (D-RSD) problems.

Notation

h denote a random oracle. λ denotes a security parameter; H_λ denotes a random oracle whose output depends on λ. Once again, given a prover-verifier protocol, l_λ denotes the number of rounds to run to achieve security related to λ.

Let q be a power of a prime p, m an integer and let V_n be an n dimensional vector space over a finite field $\mathbb{F}_{q^m}$. Let β denote a basis $(\beta_1, \ldots, \beta_m)$ of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$. Let $\mathcal{F}_i$ be the map from $\mathbb{F}_{q^m}$ to $\mathbb{F}_q$ where $\mathcal{F}_i(x)$ is the $i-$th coordinate of x in the basis β. To any $v = (v_1, \ldots, v_n) \in V_n$, we associate $\overline{v} \in \mathcal{M}_{m,n}(\mathbb{F}_q)$ defined by $\overline{v}_{i,j} = \mathcal{F}_i(v_j)$. For a basis β, we denote ψ_β the inverse of the application $V_n \to \mathcal{M}_{m,n}(\mathbb{F}_q) : x \to \overline{x}$ computed with the basis β. Here, $\omega t(x)$ denotes Rank weight of x that is defined below.

Before going into details, we give some recalls about Rank Metric.

6.2 Background on Rank metric and Cryptography

One should notice that, even if some subtle differences may appear due to Rank structure, Hamming and Rank metrics are philosophically close so that Rank-based ideas and problems are often adaptations of Hamming ones.

6.2.1 Rank Metric Codes

We first recall some definitions.

Définition 51 (Matrix Code)**.** *A linear matrix code C of length $m \times n$ over $\mathbb{F}_q$ is a subspace of matrices space of size $m \times n$ over $\mathbb{F}_q$. If C is of dimension K, we say that C is a $[m \times n, K]_q$ matrix code, or $[m \times n, K]_q$ if there is no ambiguity.*

Pursuing with notation, one can notice that each word $x = (x_1, x_2, \ldots, x_n) \in \mathbb{F}_{q^m}^n$ can be associated to a $m \times n$ matrix over $\mathbb{F}_q$ by representing each coordinate c_i by a column vector with respect to a basis β. More precisely, x can be rewritten as follows:

$$x = \begin{pmatrix} x_1 & x_2 & \ldots & x_n \end{pmatrix} \in \mathbb{F}_{q^m}^n \tag{6.1}$$

$$\overline{x} = \begin{pmatrix} x_{1,1} & x_{1,2} & \ldots & x_{1,n} \\ x_{2,1} & x_{2,2} & \ldots & x_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m,1} & x_{m,2} & \ldots & x_{m,n} \end{pmatrix} \in \mathbb{F}_q^{m \times n}, \tag{6.2}$$

where $\forall i = 1 \ldots n, x_i = (x_{1,i}, \ldots, x_{m,i}) \in \mathbb{F}_{q^m}$.

The difference between such a $[m \times n, K]_q$ matrix code and a code of length mn and of dimension K is that we can define a natural metric through the matrix rank function.

Définition 52. *For any $v \in V_n$, the rank weight of v, denoted $\omega t(v)$, is defined as the rank of the associated matrix $\overline{v}$. We can now define the rank metric between two vectors x and y such as $d_r(x, y) \stackrel{def}{=} \omega t(x-y) = \mathsf{rk}(\overline{x}-\overline{y})$. From now, $\mathcal{B}(S, \omega) \stackrel{def}{=} \{v \in S : \omega t(v) = \omega\}$.*

Définition 53 (Linear Rank Code)**.** *A $[n,k]_{q^m}$ rank code C of length n and dimension k over $\mathbb{F}_{q^m}$ is a linear subspace of dimension k of $\mathbb{F}_{q^m}^n$ viewed as a rank metric space.*

Defined as the rank of its associate matrix $\overline{x}$, the weight of a word x does not depend on the choice of the basis β.

Définition 54. *Let $x = (x_1, \ldots, x_n) \in \mathbb{F}_{q^m}^n$ be a vector of rank ω. We denote E the $\mathbb{F}_q$-subvector space of $\mathbb{F}_{q^m}$ generated by $x_1, x_2, \ldots, x_n$. The vector space E is called the* ***support*** *of x, further denoted $\mathsf{Supp}(x)$.*

Remark 5. *The notion of support of a codeword for the Hamming metric and for the Rank metric are different but share a common principle: in both cases, given a syndrome s for which it exists a low weight vector x such that $H.x^\top = s$, then, if the support of x is known, it is possible to recover all the coordinates values of x by solving a linear system.*

Bounds for Rank metric

Classic bounds for Hamming metric (almost) straightforwardly extend to Rank metric. We then give Singleton and Gilbert-Varshamov analogues.

Singleton bound For $[n,k]_{q^m}$ a linear code of rank r, we have $r \leqslant 1 + n - k$. When $n > m$, following [19], this can be rewritten as : $r \leqslant 1 + \left\lfloor \frac{(n-k)m}{n} \right\rfloor$.

Gilbert-Varshamov bound (GVR) The number of elements $S(m,q,\omega)$ of a sphere of radius ω in $\mathbb{F}_{q^m}^n$, is equal to the number of $m \times n$ q-ary matrices of rank weight t. For $t = 0$, $S_0 = 1$ and for $\omega \geq 1$, we have (see [19]):

$$S(n,m,q,\omega) = \prod_{j=0}^{\omega-1} \frac{(q^n - q^j)(q^m - q^j)}{q^t - q^j}.$$

From this, we then deduce the volume of a ball $B(n,m,q,\omega)$ of radius t in $\mathbb{F}_{q^m}$ to be:

$$B(n,m,q,\omega) = \sum_{i=0}^{\omega} S(n,m,q,i).$$

In the (frequent) linear case, the rank Gilbert-Varshamov bound $GVR(n,k,m,q)$ for a linear code $[k,n]_{q^m}$ is then defined as the smallest integer ω such as $B(n,m,q,\omega) \geq q^{m(n-k)}$ where $\mathcal{B}(n,m,q,\omega)$ denotes a ball of radius ω in in $\mathbb{F}_{q^m}$.

For a rank code C with dual matrix H, the GVR bound is the smallest rank weight ω for which, for any syndrome s, there exists on average one word x solving the RSD instance (H,s,ω).

Decoding in Rank metric

Gabidulin Codes Contrary to Hamming metric, a few code families enable unique decoding. Basically, only Gabidulin codes [18], that can be seen as Reed-Solomon analogues can do so. Defined over $\mathbb{F}_{q^m}$ for $k \leq n \leq m$, they are optimal and satisfy Singleton bound for $m = n$ and $d = n - k + 1$. They can decode up to $\lfloor \frac{n-k}{2} \rfloor$ errors.

LRPC Codes [20] Standing for Low Rank Parity Check matrix, such codes enjoy a poor structure – contrary to Gabidulin ones – so that they end up to be a well-suited candidate for rank-based cryptography.

Définition 55. *A Low Rank Parity Check (LRPC) code of rank d, length n and dimension k over $\mathbb{F}_{q^m}$ is a code defined by a $(n-k) \times n$ parity check matrix $H = (h_{i,j})$, such that all its coordinates $h_{i,j}$ belong to the same $\mathbb{F}_q$-subspace of dimension d of $\mathbb{F}_{q^m}$. We denote by $\{F_1, F_2, \ldots, F_d\}$ a basis of F.*

The general idea for decoding a word y is as follows: when the parity matrix H has rank weight small enough, the space generated by the coordinates of a syndrome $s = H.y^\top$ enables to recover the product space $P = \langle E.F \rangle$. Then, knowledge of both P and F enables to deduce E the support of the error e and finally the error e contained in y by solving a linear system. Nevertheless, LRPC codes can only decode up to $\lfloor \frac{n-k}{d} \rfloor$ so that a methodology à la CFS appears hard to reach.

RankSign signature scheme [184] addresses this issue by reaing benefits of *generalized erasures*

Définition 56. *Let e be an error vector of rank r and error support space E. We denote by generalized erasure of dimension t of an error e, a subspace T of dimension t of its error support E.*

Similarly to the Hamming case where an erasure corresponds to knowing the position of an error, this rank erasure notion is the knowledge of a subspace T of the error support E.

6.2.2 Rank-Based Cryptography

The main interest of rank-based cryptography is that for hard problems with same size of parameters, the computational complexity is higher than problems based on Hamming metric. It is then possible to generate instances of problems, with high computational complexity and with small size of keys (a few thousand bits) when such sizes are only reached with additional structure (like cyclicity) for Hamming (code-based cryptography) or Euclidean (lattice-based cryptography) distances. For more details, we refer the reader to [19, 188] and references therein.

Syndrome Decoding problem (RSD)

As in the Hamming case, the problem consists in finding a weighted constrained antecedent to a random syndrome by a dual matrix.

Définition 57 (Rank Syndrome Decoding)**.** *Let H be a $(n-k) \times n$ matrix over $\mathbb{F}_{q^m}$ $(k < n)$, $s \in \mathbb{F}_{q^m}^{n-k}$ and ω an integer. The RSD problem consists in finding a vector $x \in \mathbb{F}_{q^m}^{n}$ verifying $H.x^\top = s$ and $\omega t(x) \leq \omega$.*

The RSD problem can be seen as a rank adaptation of the well-known Syndrome Decoding (SD) problem which relies on Hamming metric and was proven to be NP-complete in [57]. Via a probabilistic reduction to Hamming metric, this problem was proven hard in [189] while the complexity of the best known attacks can be found in [190].

New rank-based problems

We now introduce two rank-based problems referred as the One-More Rank Syndrome (OMRSD) Decoding problem and the Decision Rank Syndrome Decoding (D-RSD) problem. These two problems consist in adaptations of Hamming-based problems respectively introduced in Chapter 5 of this dissertation and in [186]. We prove that the D-RSD problem is a hard problem and we justify the very likely difficulty of the OMRSD problem.

One More Rank Syndrome Decoding problem We first discuss the situation where one is given some solutions of a RSD instance and is asked to find a new one. The only difference with the OMSD is the underlying metric.

Définition 58 (OMRSD problem)**.** *Given an RSD instance $sd = (H, s, \omega)$ and l solutions to sd, denoted $x_1, \ldots, x_l$, the OMRSD$(sd, x_1, \ldots, x_l)$ problem consists in finding x_{l+1} solution of sd such as : $\forall\, i = 1 \ldots l, x_i \neq x_{l+1}$.*

Assumption 1 : the OMRSD problem is hard.

Discussion on assumption 1: There is no known reduction to the RSD problem for the OMRSD problem. This problem is an adaptation in a coding context of a similar problem which exists for classical cryptography. At the difference of the classical RSD problem where an attacker knows only a syndrome and wants to find a small weight vector, in that case the attacker knows l small weight vectors of weight ω and search for a new one. It is of course possible to consider linear combinations of small weight vectors to find another small weight vector, meanwhile because of the properties of the metric, adding two random small weight vectors of weight ω leads in general to a vector of weight close to 2ω which is of no use for our problem. In particular in the case of rank metric, if ω

is greater than the rank Gilbert-Varshamov bound (which has to be the case in general, if more than one preimage of a syndrome does exist), the problem of finding a pre-image of weight more than twice the GVR bound is always easy. Hence this means that using linear combinations of known solutions is not likely to be of any help. This type of result is not true for instance for Hamming metric or for Euclidean norm, for which in some cases finding preimage of weight twice the Gilbert-Varshamov bound can be difficult. Moreover the number of linear independent such solutions is upper bounded by the dimension of the code. Overall, although there is no known reduction for this problem, the problem is considered difficult by the communauty and no attack exploiting the l known vectors are known, so that the best attack for the problem consists in directly attacking the RSD problem.

Decision Rank Syndrome Decoding problem We now define the D-RSD problem which consists in distinguishing a random syndrome from a syndrome issued from a small weight vector.

Définition 59 (D-RSD problem)**.** *Given a random* $H \in \mathcal{M}_{n-k,n}(\mathbb{F}_{q^m})$*, a word* $x \in \mathcal{B}(\mathbb{F}_{q^m}^n, \omega > 0)$ *a random syndrome* $s \in \mathbb{F}_{q^m}^{n-k}$*, is it possible to distinguish* $H.x^\top$ *from* s*?*

Once again, this problem can be seen as a rank adaptation of the Decision Syndrome Decoding problem defined in Hamming-based cryptography.

Proposition 1. *The D-RSD problem is hard.*

Proof. Decision problems are very important in cryptography; in the case of Hamming-based cryptography, the Decision Syndrome Decoding problem has been proven equivalent to the search problem in [186, Theorem 2], based on the Goldreich-Levin theorem. The result is presented in term of indistinguishability of a pseudo-random generator based on the SD problem. Recently a transformation from a binary code to a q-ary code was proposed in [189] which permits to obtain a randomized reduction from the SD problem to the RSD problem. This transformation was used in [191] to adapt the result of Fisher-Stern [186] for rank metric, but with a reduction to the computational SD problem. These results hence show that there is a randomized reduction from the binary computational SD problem to the D-RSD problem, and hence that the D-RSD problem is hard. In practice the best attacks for this problem are attacks towards the RSD problem. □

Rank Stern-like protocol

Introduced by Goldwasser, Micali and Rackoff [55], Zero-Knowledge (ZK) protocols fast aroused interest. Stern then first proposed such a scheme based on coding theory [162].

Fixing the attempt of Chen [192], Gaborit *et al.* designed a 3-pass prover-verifier protocol constituting a rank alternative to Stern's protocol [185]. To fulfill such a goal, they had to define, in rank metric, an equivalent notion of permutation used in the Hamming metric setting. More precisely, they came up with an operation that, without leaking any information about its support, can associate any word of rank ω to any particular word of same rank ω. We now recall this operation.

Definition 14. *Let $Q \in GL_m(q)$, $v \in V_n$ and a basis β. We define the product $Q * v$ such that $Q * v \stackrel{def}{=} \psi_\beta(Q\overline{v})$. For any $x, y \in V_n$ such that $rk(x) = rk(y)$, it is possible to find $P \in GL_n(q)$ and $Q \in GL_m(q)$ such that $x = Q * yP$.*

This operation enables a straightforward conversion from Hamming metric (Figures 5.1) to Rank metric.

6.2.3 LRPC related cryptosystems

Since LRPC are poorly structured with efficient decoding algorithm [20], they are successfully for cryptographic purposes.

LRPC Cryptosystem

When embedded into either McEliece or Niederreiter cryptographic setting (see Chapter 1, Section 1.4), they enable the design of an LRPC-based encryption scheme. In Section 6.7), we will focus on the Niederreiter setting to instantiate our scheme. Following recalls of 1.3, such an encryption is defined as follows:

- $\mathsf{KeyGen}(1^\lambda)$: generates $\mathcal{C}$ an LRPC code of support $\mathcal{S}$ and rank r. Its parity check matrix is denoted $H \in \mathbb{F}_{q^m}^{(n-k)\times n}$ with corresponding generator matrix $G \in \mathbb{F}_{q^m}^{k\times n}$. Sets and returns $\mathsf{pk} = (G, r)$ et $\mathsf{sk} = H$
- $\mathsf{Encrypt}(\mathsf{pk}, x)$: randomly generates $e \xleftarrow{\$} \mathbb{F}_{q^m}^n$ of rank $\omega(e) \leq r$ to return $c = xG + e$.
- $\mathsf{Decrypt}(\mathsf{sk}, c)$: computes $s = Hc^\top$; uses sk to decode s and recover e. Finally sets $xG = c - e$ to recover x.

RankSign or Decoding a random syndrome beyond GVR [183]

The traditional approach for decoding random syndromes, that is used by the CFS scheme for instance, consists in taking advantage of the decoding properties of a code (e.g. a Goppa code) and considering parameters for which the proportion of decodable vectors – the decodable density – is not too low. For the Hamming metric, this approach leads to very flat dual matrices, *i.e.*, codes with high rate and very low Hamming distance. In the

rank metric case, this approach leads to very small decodable densities and does not work in practice. However, Gaborit et al. [183, 184] then proposed to proceed otherwise. It turns out that the decoding algorithm of LRPC codes can be adapted so that it is possible to decode not only errors but also (generalized) erasures. This new decoding algorithm allows us to decode more rank errors since the support is then partially known. In that case since the size of the balls depends directly on the dimension of the support, it leads to a dramatic increase of the size of the decodable balls.

The RankSign signature scheme was then deduced from this result applying a methodology close to CFS where, given a secret key, one is then able to output a small weight vector solving an RSD instance relatively to a public matrix. The RankSign public key can then be seen as a trapdoor matrix.

We now define the group signature instantiated in this work.

6.3 Definition and Security Model

This section is dedicated to our new group signature definition. Contrary to the work presented in Chapter 5, this scheme does not enable users to issue their own secret keys.

6.3.1 Definition

Under the existence of a PKI for exchanges between users and authorities, we propose the following definition where two authorities, a group manager (also called issuer) and an opener, are involved.

Definition 15. *A group signature $\mathcal{GS}$ scheme is a sequence of protocols* (KeyGen, Join, Sign, Verif, Open) *such as:*

- $\mathsf{KeyGen}(1^\lambda)$*: it generates the group public key* gpk *and the private keys: the group manager secret key* gmsk *and the opener secret key* skO *containing some tracing table* tr *which could be publicly revealed;*
- $\mathsf{Join}(\mathcal{U}_i, \mathsf{gmsk}, \mathsf{gpk})$*: interactive protocol between a user* $\mathcal{U}_i$ *and the group manager. In the end, the user gets a secret key* $\mathsf{usk}[i]$ *and the issuer contacts the opener to update* tr*;*
- $\mathsf{Sign}(\mathsf{usk}[i], \mathsf{gpk}, m; \mu)$*: to sign a message* m*, the user uses his secret key* $\mathsf{usk}[i]$ *and some randomness* μ *to output a signature* σ *valid under the group public key;*
- $\mathsf{Verif}(\mathsf{gpk}, m, \sigma)$*: anybody can check the validity of a signature* σ *on the message* m *with respect to* gpk*. It outputs* 1 *if the signature is valid, and 0 otherwise;*

- $\mathit{Open}(\mathit{skO}, \mathit{gpk}, m, \sigma)$*: for a valid signature* σ *with respect to* gpk*, the opener can provide signer's identity: it thus outputs the user* $\mathcal{U}_i$ *when it succeeds and 0 otherwise.*

6.3.2 Security Model

Like our Hamming-based construction, our scheme is dynamic (protocol *Join*) but does not fulfill the non-frameability security property required by the dynamic BSZ model. Indeed, since users are given their secret keys -while the work of Chapter 5 enabled them to issue half of it-, they cannot prevent aut do not have control on their secret keys

Remark 6. *Informally, non-frameability guarantees that, even if both the group manager and the opener are corrupted, no honest user could be accused of having generated a signature if he did not. Even if non-frameability appears as a nice property, many real life applications, such as authentication systems, assume issuer integrity so that the interest of our model does hold in numerous contexts.*

We then require our scheme to fulfill properties of correctness, anonymity and traceability.

Correctness

This guarantees that honest users should be able to generate valid signatures, and the opener should then be able to revoke anonymity of the signers.

In the following experiments, we denote the set of corrupted users by CU, made of users for which an adversary $\mathcal{A}$ knows their secret keys in opposition to honest users referred as the set HU. $\mathcal{A}$ is granted some oracles:

- $\mathcal{O}^{\text{join}}(\mathcal{U}_i)$, a new user $\mathcal{U}_i$ is added to HU;
- $\mathcal{O}^{\text{sign}}(\mathcal{U}_i, m)$, if $\mathcal{U}_i \in \mathsf{HU}$, returns $\mathsf{Sign}(\mathsf{gmsk}, \mathsf{sk}[i], m)$ and adds i to $S[m]$, the list of users for which a signature on message m exists;
- $\mathcal{O}^{\text{corrupt}}(\mathcal{U}_i)$, if $\mathcal{U}_i \in \mathsf{HU}$, provides user's secret key $\mathsf{usk}[i]$ and moves $\mathcal{U}_i$ to CU;
- $\mathcal{O}^{\text{open}}(m, \sigma)$, returns $\mathsf{Open}(\mathsf{skO}, \mathsf{gpk}, m, \sigma)$.

Anonymity and Traceability

We recall that anonymity requires that signatures issued by two users are computationally indistinguishable to an adversary $\mathcal{A}$ while traceability ensures that no group member or coalition of group members and the opener can produce a valid signature that cannot be opened or for which the opening process might accuse an honest user. Botj of these notions were introduced in previous chapter but we recall them for the sake of clarity.

Anonymity The anonymity security game (Figure 6.1 (a)) consists in a challenger randomly choosing a bit $b \in \{0,1\}$ while the adversary $\mathcal{A}$ is asked to guess this value. More precisely, $\mathcal{A}$ targets two users i_0 and i_1 and the challenger issues a signature on behalf of i_b. Granted aforesaid oracles, the adversary wins the game if it outputs $b' = b$.

Traceability Concerning traceability (Figure 6.1 (b)), the adversary aims at producing a valid signature for which the opening procedure either fails or accuses an honest user. More precisely, algorithm *Verif* must output 1 on inputs a cople (m, σ) generated by the adversary and gpk while procedure *Open* should not output the identity of a corrupted user because it would simply mean that $\mathcal{A}$ signed m with a secret key he already knew. On the contrary, if $\mathcal{A}$ is able to produce a valid signature that cannot be opened or that traces back to an honest user, we consider that he has succeeded in attacking the security of the scheme.

(a) Experiment $Exp^{anon-b}_{\mathcal{GS},\mathcal{A}}(\lambda)$
1. $(\mathsf{gpk}, \mathsf{gmsk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$
2. $(m, i_0, i_1) \leftarrow \mathcal{A}(\mathsf{gpk}, \mathsf{tr} : \mathcal{O}^{\text{join}}, \mathcal{O}^{\text{corrupt}}, \mathcal{O}^{\text{sign}}, \mathcal{O}^{\text{open}})$
3. $\sigma_b \leftarrow \mathsf{Sign}(\mathsf{usk}[i_b], \mathsf{gpk}, m; \mu)$
4. $b' \leftarrow \mathcal{A}(\mathsf{gpk}, \sigma_b : \mathcal{O}^{\text{join}}, \mathcal{O}^{\text{corrupt}}, \mathcal{O}^{\text{sign}}, \mathcal{O}^{\text{open}})$
5. If $i_0 \notin \mathsf{HU}$ or $i_1 \notin \mathsf{HU}$, Return 0.
6. Return b'.

$$Adv^{anon}_{\mathcal{GS},\mathcal{A}}(\lambda) = Pr[Exp^{anon-1}_{\mathcal{GS},\mathcal{A}}(\lambda) = 1] - Pr[Exp^{anon-0}_{\mathcal{GS},\mathcal{A}}(\lambda) = 1]$$

(b) Experiment $Exp^{tr}_{\mathcal{GS},\mathcal{A}}(\lambda)$
1. $(\mathsf{gpk}, \mathsf{gmsk}, \mathsf{skO}) \leftarrow \mathsf{KeyGen}(1^\lambda)$
2. $(m, \sigma) \leftarrow \mathcal{A}(\mathsf{gpk}, \mathsf{skO} : \mathcal{O}^{\text{join}}, \mathcal{O}^{\text{corrupt}}, \mathcal{O}^{\text{sign}})$
3. If $\mathsf{Verif}(\mathsf{gpk}, m, \sigma) = 0$, Return 0.
4. If $\mathsf{Open}(\mathsf{skO}, \mathsf{gpk}, m, \sigma) = \perp$, Return 1.
5. If $\exists j \notin \mathsf{CU} \cup S[m]$,
   $\mathsf{Open}(\mathsf{skO}, \mathsf{gpk}, m, \sigma) = j$, Return 1.
6. Else Return 0.

$$Adv^{tr}_{\mathcal{GS},\mathcal{A}}(\lambda) = Pr[Exp^{tr}_{\mathcal{GS},\mathcal{A}}(\lambda) = 1]$$

Figure 6.1: Security notions

Definition 16. *A group signature scheme fulfilling correctness and for which advantages related to anonymity and traceability (Figure 6.1) are negligible, is said to be dynamic.*

6.4 Rank Concatenated Stern's protocol

For H a public matrix, x a small weight vector of weight w_x, and the syndrome $s = H.x^\top$, Stern's authentication protocol [162] permits a prover to convince a verifier that he knows a small weight vector of weight w_x, such that $H.x^\top = s$. We saw, notably in Chapter 5,

that Stern's authentication protocol and its variations have been widely used to design group signatures through FS paradigm. A rank-based alternative was first proposed by [192] that was later broken and repaired by Gaborit et al. in [185]. We rely on this latter, referred as the *Rank Stern's protocol* to propose a new rank-based ZK authentication protocol.

6.4.1 Problematic and Overview of our protocol

Since the problematic tackled is a straight adaptation of the Hamming case discussed in previous chapter, we refer the reader to subsection 5.3.1 for more details. It only differs by the metric and the ensuing tools. More precisely, let us consider $k \times n$ and $k \times n'$ random matrices Q and R over $\mathbb{F}_{q^m}$, a syndrome s, some weights ω_x and ω_y leading to the SD instance depicted in Figure 6.2.

$$\begin{bmatrix} Q & | & R \end{bmatrix} . \begin{pmatrix} x \\ y \end{pmatrix} = s.$$

Figure 6.2: High level overview

6.4.2 Rank Concatenated Stern's protocol ($\mathcal{RCSP}$)

Similarly to previous Chapter (Figure 5.2), this protocol, from now referred as *Rank Concatenated Stern's Protocol* ($\mathcal{RCSP}$), aims to prove knowledge of a secret (x, y) with weight constraints on both x and y. The idea is somewhat to run, in parallel, 2 instances of Rank Stern's protocol on x and y while linking these two values through commitments. We denote here $V_n = \mathbb{F}_{q^m}^n$ and $V_{n'} = \mathbb{F}_{q^m}^{n'}$.

Remark 7. *Contrary to previous chapter where the TwZK notion was involved, we make use of randomized commitments here (roles of seeds r_1, r_2, r_3) to fulfill ZK property.*

Theorem 7. *$\mathcal{RCSP}$ (Figure 6.3) is an honest prover-verifier ZK protocol with cheating probability $2/3$ thus verifying properties of completeness, soundness and zero-knowledge.*

Proof. Lying on rank version of Stern's protocol [185], the proof is straightforward. Completeness property is straightforward and we only stress out that in the case where $\mathsf{ch} = 1$, the verifier can check validity of c_1 by computing $h(Q_1|P_1|(Q(v_1+x)^\top + R(v_2+y)^\top - s|Q_2|P_2|r_1)$. Soundness and ZK properties directly come from those (of the randomized) Rank Stern's protocol. □

RSD instance $((Q|R), s, \omega_Q, \omega_R)$
$\mathcal{P}$'s secret: $(x, y) \in V_n \times V_{n'}$ such as $(Q|R).(x, y)^\top = s$ with $\omega t(x) = \omega_x$ and $\omega t(y) = \omega_y$

1. [Commitment step] $\mathcal{P}$ chooses $(v_1, v_2) \xleftarrow{\$} V_n \times V_{n'}$, $r_1, r_2, r_3 \xleftarrow{\$} 1^\lambda$,
   $(P_1, P_2) \xleftarrow{\$} GL_n(\mathbb{F}_q) \times GL_{n'}(\mathbb{F}_q)$ and $Q_1, Q_2 \xleftarrow{\$} GL_m(q)$ and
   He then sends c_1, c_2, c_3 where:
   $c_1 = h(Q_1|P_1|Qv_1^\top + Rv_2^\top|Q_2|P_2|r_1)$,
   $c_2 = h(Q_1 * v_1P_1|Q_2 * v_2P_2|r_2)$,
   $c_3 = h(Q_1 * (v_1 + x)P_1|(Q_2 * (v_2 + y)P_2|r_3)$
2. [Challenge step] $\mathcal{V}$ sends $\mathsf{ch} \xleftarrow{\$} \{0, 1, 2\}$ to $\mathcal{P}$.
3. [Response step] There are three possibilities:
   $\mathsf{ch} = 0$: $\mathcal{P}$ responds v_1, $(Q_1|P_1)$, v_2, $(Q_2|P_2)$, r_1, r_2.
   $\mathsf{ch} = 1$: $\mathcal{P}$ responds $v_1 + x$, $(Q_1|P_1)$, $v_2 + y$, $(Q_2|P_2)$, r_1, r_3.
   $\mathsf{ch} = 2$: $\mathcal{P}$ responds $Q_1 * v_1P_1$, $Q_1 * xP_1$, $Q_2 * v_2P_2$, $Q_2 * yP_2$, r_2, r_3.
4. [Verification step] There are three possibilities:
   $\mathsf{ch} = 0$: $\mathcal{V}$ checks c_1, c_2.
   $\mathsf{ch} = 1$: $\mathcal{V}$ checks c_1, c_3.
   $\mathsf{ch} = 2$: $\mathcal{V}$ checks c_2, c_3 and
   $\omega_t(Q_1 * xP_1) = \omega_x,\ \omega t(Q_2 * yP_2) = \omega_y$.
5. [Final step] $\mathcal{V}$ outputs *Accept* if all checks passed,
   $\perp$ otherwise.

Figure 6.3: Rank Concatenated Stern's Protocol ($\mathcal{RCSP}$).

6.5 Our Rank-Based Group Signature Scheme

Similarly to Chapter 5, we make use of a trapdoor matrix to find a small weight antecedent of a given syndrome. Nevertheless, because Rank metric does not maintain RSD hardness when the weight constraint is around twice GV, we cannot adopt the same joining methodology. Before going into details, we introduce matrices H_s and H_c, indistinguishable from random ones, verifying:

- H_s is a public *trapdoor matrix* i.e. given a (trapdoor) secret key sk_s, a random syndrome s and an integer ω_s, one can output y solving the RSD instance (H_s, s, ω_s);
- H_c is the public key of a Rank-based Public Key Cryptosystem (R-PKC) with associated secret key sk_c.

6.5.1 High Level Overview of our scheme

The main idea is to instantiate $\mathcal{RCSP}$ with particular matrices Q and R so that a user will be given a small weight split secret (x, y_i) such as:

- y_i is user's signing key committed into a group public syndrome s relatively to H_s;
- x is a random vector committed through a R-PKC ciphertext c enabling to further revoke anonymity.

These secrets are then linked via a syndrome r, leading to the situation depicted in Figure 6.4, where A and B are random matrices, H_s is a trapdoor matrix and H_c, a R-PKC public matrix.

$$\left[\begin{array}{c|c} A & B \\ 0 & H_s \\ H_c & 0 \end{array}\right] . \begin{pmatrix} x \\ y_i \end{pmatrix} = \begin{pmatrix} r \\ s \\ c \end{pmatrix}.$$

Figure 6.4: A particular instantiation of $\mathcal{RCSP}$.

To sign a message, $\mathcal{U}_i$ then makes a proof of knowledge on (x, y_i) through Rank Concatenated Stern's protocol (Figure 3). When the opener, given the R-PKC secret key, wants to revoke anonymity, he first recovers x from c and then computes $r - Ax^\top$. This value must appear in its tracing table tr containing all the $B.y_i^\top$s from which he can finally deduce signer's identity.

6.5.2 Algorithms KeyGen, Join and Sign

To begin, the algorithm KeyGen, according to λ, generates the following data:

- a RSD instance (H_s, s, ω_s) where H_s is a trapdoor matrix with associated secret key sk_s given to the group manager;
- a R-PKC key pair (H_c, sk_c) with sk_c given to the opener;
- an integer ω and two random matrices A, B.

When contacted by user $\mathcal{U}_i$, the group manager uses its trapdoor key sk_s to compute user's secret key $\mathsf{usk}[i] = y_i$ as a solution of the aforesaid RSD instance. The opener is then given the syndrome $B.y_i^\top$ to update tr.

Now, to authenticate himself, $\mathcal{U}_i$ first chooses a random x of weight ω and computes the syndromes $r = A.x^\top + B.y_i^\top$ and $c = H_c.x^\top$. By instantiating $Q = \left[\begin{smallmatrix} A \\ 0 \\ H_c \end{smallmatrix}\right]$ and $R = \left[\begin{smallmatrix} B \\ H_s \\ 0 \end{smallmatrix}\right]$, he makes, through $\mathcal{RCSP}$, a ZK proof on the secret (x, y_i) with $\omega t(x) = \omega$ and $\omega t(y_i) = \omega_s$, solving the RSD instance depicted in Figure 6.4.

Finally, by turning this process non-interactive through FS paradigm, we get the signing algorithm. Algorithms KeyGen, Join and Sign are described in Figure 6.5.

6.5.3 Algorithms Verif and Open

The verification algorithm relies on the verification step of $\mathcal{RCSP}$ (Figure 6.3).

To revoke anonymity, the opener uses sk_c to recover x from the R-PKC ciphertext $c = H_c.x^\top$. He can then computes $A.x^\top$ and subtracts this value to r transmitted in the

(a) $\mathsf{KeyGen}(1^\lambda)$
1. According to λ, generate:
   1.1. a RSD instance $\mathsf{rsd} = (H_s, s, \omega_s)$:
      – H_s a trapdoor matrix;
      – sk_s its related secret key.
   1.2. a R-PKC instance:
      – H_c the public matrix key;
      – sk_c its related secret key.
   1.3. two random matrices A and B and an integer ω.
4. $\mathsf{gpk} := (H_s, H_c, A, B, s, \omega_s, \omega)$.
5. $\mathsf{gmsk} := \mathsf{sk}_s$, $\mathsf{skO} := (\mathsf{sk}_c, \mathsf{tr} = [\,])$.
6. Return $(\mathsf{gpk}, \mathsf{gmsk}, \mathsf{skO})$.

(b) $\mathsf{Join}(\mathcal{U}_i, \mathsf{gmsk} = \mathsf{sk}_s, \mathsf{gpk})$
1. Use the trapdoor sk_s on H_s to output y_i solving rsd.
2. If $\exists\, j \leq i-1$, $y_i = \mathsf{usk}[j] \vee B.y_i^\top = \mathsf{tr}[j]$, go to 1.
3. Return $\mathsf{usk}[i] = y_i$, $\mathsf{tr}[i] = B.y_i^\top$.

(c) $\mathsf{Sign}(\mathsf{usk}[j] = y_i, \mathsf{gpk}, m; \mu)$
1. Choose a random x such as $\omega t(x) = \omega$.
   1.1. $r := A.x^\top + B.y_i^\top$.
   1.2. $c := H_c.x^\top$.
2. For $l = 1 \ldots l_\lambda$
   2.1. Set $c_1, c_2, c_3, d_1, d_2, d_3$ according to Figure 3, step 1.
   2.2. $\mathsf{cmt}[l] := \{c_1, c_2, \ldots, d_3\}$.
3. $\mathsf{ch} := \mathsf{H}_\lambda(m, \mathsf{cmt}, r, c) \in [\![2]\!]^{l_\lambda}$.
4. For $i = 1 \ldots l_\lambda$
   Generate $\mathsf{rsp}[l]$ according to $\mathsf{ch}[l]$ and Figure 3, step 3.
5. Set $\Pi = (\mathsf{cmt}, \mathsf{ch}, \mathsf{resp})$.
6. Return $\sigma = (\Pi, (r, c))$.

Figure 6.5: KeyGen, Join, Sign algorithms

signature along with c. The result $r - A.x^\top$ must be equal to some $\mathsf{tr}[i]$, from which the signer's identity is learnt. These two algorithms appear in Figure 6.6.

(a) $\mathsf{Verif}(\mathsf{gpk}, m, \sigma)$
1. Parse $\sigma = (\Pi, (r, c))$
2. Parse $\Pi = (\mathsf{cmt}, \mathsf{ch}, \mathsf{rsp})$
3. $\mathsf{ch}^* := \mathsf{H}_\lambda(m, \mathsf{cmt}, r, c) \in [\![2]\!]^{l_\lambda}$. If $(\tilde{\mathsf{ch}} \neq \mathsf{ch})$, Return 0.
4. For $l = 1 \ldots l_\lambda$
   3.1. Check $\mathsf{rsp}[l]$ according to $\mathsf{cmt}[l], \mathsf{ch}[l]$ and Figure 3.
   3.2. If a verification fails, Return 0.
5. Return 1.

(b) $\mathsf{Open}(\mathsf{skO}, \mathsf{gpk}, m, \sigma)$
1. If $(\mathsf{Verif}(\mathsf{gpk}, m, \sigma) = 0)$ Return $\perp$.
2. Parse $\sigma = (\Pi, (r, c))$
3. Parse $\mathsf{skO} = (\mathsf{sk}_c, \mathsf{tr})$.
4. Use sk_c to recover x from c.
5. Set $z = r - Ax^\top$
6. For $i = 1 \ldots N$
   If $(\mathsf{tr}[i] = z)$ Return $\mathcal{U}_i$.
7. Return $\perp$.

Figure 6.6: Verif and Open algorithms

6.6 Security Analysis

Since correctness directly comes from $\mathcal{RCSP}$, we focus on anonymity and traceability requirements defined in Figure 6.1. We begin with the anonymity property.

Theorem 8. *If there exists an adversary $\mathcal{A}$ that can break the anonymity property of the scheme, then there exists an adversary $\mathcal{B}$ that can either break the Zero-Knowledge property of $\mathcal{RCSP}$ or the Decision Rank Syndrome Decoding (D-RSD) problem.*

Proof. Through a sequence of games, we will exhibit that an adversary against our anonymity property would be able to either break the ZK property of our scheme or the D-RSD problem.

- $\mathcal{G}_0$ $\mathcal{B}$ runs $\mathsf{KeyGen}(1^\lambda)$ and acts honestly as described in Figure 6.1 (a).
- $\mathcal{G}_1$ Now, to answer the opening query, the simulator uses the ROM observability to extract some y, and then compares the value to the $B.y_i^\top$ contained in tr. Under the ZK soundness, this is similar to the previous game.
- $\mathcal{G}_2$ The simulator now supersedes part of the KeyGen by setting $\begin{bmatrix} A \\ H_c \end{bmatrix} = C$. This game is identical to the previous one.
- $\mathcal{G}_3$ The simulator now simulates the proof when answering the challenge queries, by not using the value x, y_i. This game is identical to the previous one under the ZK property.
- $\mathcal{G}_4$ Now, he sends randoms $c = s_1$, $r = s_2 + B.y_i^\top$. This game is indistinguishable from the previous one under the D-RSD problem. (This is seen, by splitting the challenge s in s_1, s_2 as it was done for the matrix C)
- $\mathcal{G}_5$ This last game only displays random values, hence the adversary has no advantage, which terminates the proof.

□

Concerning traceability, we have the following theorem.

Theorem 9. *If there exists an adversary $\mathcal{A}$ that can break the traceability of the scheme, then there exists an adversary $\mathcal{B}$ that can break either break the Soundness of the Zero-Knowledge proof or the OMRSD problem.*

Proof. The proof is straightforward, the simulator starts by simulating the ZK proofs on every honest signing queries. Then, he picks an identity at random expecting it to be the targeted honest user (this happens with non-negligible probability) and sets his tracing

key $B.y_*^\top$ as s, for the other identities the simulator sends a request to the RSD oracle, and forwards the answer.

Receiving the adversary answers, the simulator uses the ROM, to extract the value y_* solution to the challenge.

□

6.7 Instantiation

Our scheme is generic and can be used with any trapdoor matrix and public key encryption scheme. In our rank-based context, RankSign and LRPC embedded into Niederreiter setting (see Section 6.2.3) constitute well-suited candidates to respectively instantiate matrices H_s and H_c introduced in Section 6.5.

According to Section 6.5 and 6.6, correctness and security of our scheme generically rely on matrices H_s and H_c meant to be indistinguishable from random ones. With such an instantiation, the security is maintained through the putative indistinguashibilty of LRPC and RankSign public matrices with random matrices [20].

Parameters

As it will exhibited below, it is sufficient in practice to consider matrices A and B with only one row and then according to previous section, the security of the protocol relies on the D-RSD problem or the OMRSD problem associated to matrices H_s and H_c. We now give parameters to obtain an overall security of 2^{100}.

Following [183, 184], we can consider parameters $n' = 23$, $k' = 10$, $t = 3$, $m = 24$ and $q = 2^8$ to design a RankSign public matrix H_s of dimension $(n' - k') \times (n' + t)$ with coordinates lying in $\mathbb{F}_{q^m}$. In particular, t denotes the number of generalized erasures handled by such an instantiation. The size of the group then consists in the number of potential antecedents to a common public syndrome s; namely it corresponds to the number of possibilities to form t independent vectors lying over $\mathbb{F}_{q^m}$ which roughly leads to q^{tm} users ($2^{8\times 24}$ here). We refer the reader to [183, 184] for more details on this point. On the other hand, the matrix H_c can be instantiated with the cyclic LRPC cryptosystem embedded in Niederreiter setting; following [20], we consider parameters $n = 74$, $k = 37$, $q = 2^8$ and the working field $\mathbb{F}_{q^m}$. Now, matrices A and B are only used to differentiate the $B.y_i^\top$ (procedure Open) and since $q = 2^8$ and $m = 24$, there are 2^{192} possibilities for $B.y_i^\top$ by simply taking A and B with one row. Finally, with A and B made of one row and H_c cyclic, the size of the public key is mainly due to H_s. By considering the systematic form with aforesaid parameters, it leads to H_s of size $(n' - k') \times (n' + t - n' + k') = (23 - 10) \times (10 + 3)$ over $GF(2^{8\times 24})$. Adding contributions

of A, B and one line of H_c, it finally leads to a public key of around 5 kB.

The signature size depends both on the security level and the length of a proof of knowledge in $\mathcal{RCSP}$ (Figure 6.3). Let us first notice that random matrices involved in the protocol can be sent through seeds from which they could be regenerated. Hence, the preponderant data sent during the protocol consists in the vectors belonging to V_n and $V_{n'}$: thus we get on average 4/3 elements of the ambient space $\mathbb{F}_{q^m}^{n+n'}$ representing $4/3\times(74+23)\times 8\times 24$ bits. When targeting a 100 bits security level for which $100/log_2(3/2)$ repetitions of the protocol are required, this leads to a signature of 550 kB. One should notice that these parameters are versatile and it would be easy to find parameters to fit another security level.

Asymptotic complexity

To study the asymptotic complexity, we first recall that:

- the number of users N is roughly $q^{mt} = 2^{mtlog(q)}$;
- in practice and as exhibited above, parameters t, k' and m are set to $O(n')$ so that the public key is approximated by $O(n'^3 \times log(q))$.

For a given security level, it is possible to increase the number of users by increasing the size of q. In that case we consider all parameters except q as fixed. From the previous recalls, the number of users is then $N = 2^{O(log(q))}$ when the size of parameters is in $O(log(q)) = O(log(N))$.

Finally, in terms of computation time, the protocol is very efficient since in themselves the LRPC and RankSign cryptosystems are very fast (a few milliseconds for encryption/decryption or signature).

Concurrent Works

Our dynamic scheme compares very well with code-based concurrent works such as [6, 176]. Indeed, it features public key and signature sizes logarithmic in N while those of the static scheme presented in [176] are linear in N. Furthermore, when considering at maximum 2^{24} users, the latter one leads to a public key of size 1.16 GB with the advantage of only relying on the SD-problem. Even if dynamic and with public key and signature sizes in $N^{1/\sqrt{log(N)}}$, the group signature of [6] leads to signatures of size 20 MB and a public key of 2.5 MB. In parallel, despite recent progress and satisfying asymptotic performances [172, 174, 175, 177, 187, 182] (public keys and signatures logarithmic in the number of group members), lattice-based constructions still suffer great sizes of parameters. Indeed, the most efficient one due to [187], improving works of [174, 175], proposes signatures and a public key respectively of size 61.5 MB and 4.9 MB for a group made of only 1024

users and an overall security of 2^{80}. Figure 6.7 sums up the situation and even mentions pairings to give a fair and overall comparison.

	Security	$N <$	Public key	Signature	Complexity in N
Lattices [187]	80b	1024	4.9MB	61.5MB	$O(log(N))$
Codes [176]	80b	2^{24}	1.16GB	196MB	$O(N)$
Chapter 5 [6]	80b	2^{400}	20MB	2.5MB	$O(N^{1/\sqrt{log(N)}})$
Pairings [166]	80b	2^{80}	**1kB**	**2kB**	$O(log(N))$
This chapter [7]	**100b**	2^{192}	**5kB**	**550kB**	$O(log(N))$

Figure 6.7: Comparison with concurrent schemes

6.8 Conclusion

Dans ce chapitre, nous avons proposé le premier schéma de signature groupe basée sur la métrique Rang. En plus d'être dynamique, il est résistant à l'ordinateur quantique et bénéficie de complexités logarithmiques en le nombre d'utilisateurs, menant à des tailles de clés (550 kB) et de signature (5 kB) surpassant tous ses concurrents post-quantiques et approchant même certaines constructions basées sur les couplages bilinéaires. Bien que reposant sur des problèmes pour lesquels il n'existe ucune attaque, la sécurité de notre construction est conditionnée à celles des problèmes sous-jacents (OMRSD, RankSign) qui, à l'heure actuelle, ne jouissent que d'une sécurité putative.

En reprenant nos considérations liées au projet TA, une limitation serait que le schéma actuel ne permet à un utilisateur de décider (d'une partie) de la valeur de sa clé de signature contrairement aux travaux du Chapitre précédent. D'ailleurs, c'est précisément ce point qui nous permettait d'obtenir la propriété de *non-frameability* que nous n'avons pas ici (Remarque 6). Ainsi, Chapitres 5 et 6 présentent respectivement :

- Un schéma basé sur la métrique de Hamming qui répond théoriquement à la limitation L9 mais pour lequel les instanciations proposées ne sont pas satisfaisantes;
- Un schéma basé sur la métrique Rang qui propose la meilleure instanciation actuelle dans le domaine post-quantum mais qui souffre d'une sécurité supposée (OMRSD, RankSign) et d'une incompatibilité philosophique avec nos travaux où un utilisateur/terminal doit participer à la génération de ses propres clés secrètes.

Finalement, les Chapitres 5 et 6 enrichissent la littérature avec des avantages certains (sécurité prouvée pour la métrique de Hamming et tailles de clés réduites dans le cas de la métrique rang) mais leurs réalisations ne répondent que partiellement à la limitation L9.

Conclusion

Avec comme toile de fond le projet TA, nous nous sommes principalement intéressés aux domaines des Fuzzy Extractors et des schémas de signature de groupe au cours de cette thèse.

Dans un premier temps et afin de répondre aux limitations pressantes du projet TA, nous avons proposé le module FAST qui, notamment par l'usage de Fuzzy Extractors (Dodis *et al.*, Eurocrypt 2004), permet d'authentifier un utilisateur via son smartphone à partir de données variées et variables (IMEI, IMSI, hachés de librairies, liste de contacts, liste d'applications, ...). Une telle réalisation a donc renforcé et amélioré l'expressivité de la solution TA dont l'objectif premier est de fournir une solution d'authentification devant satisfaire certains requis (s'affranchir d'élément matériel, assurer le respect de la vie privée, ...) décrits à la Figure 2.8.

La conception du module FAST a renforcé la nécessité de manipuler des Fuzzy Extractors réutilisables. Alors que le résultat récent de Canetti *et al.* (Eurocrypt 2016), dont la sécurité repose sur le modèle de l'oracle aléatoire, convient dans le cas de la distance de Hamming et pour la différence d'ensembles dans le cas du *petit univers*, la plupart des attributs que nous avons considérés sont en fait adaptés à la différence d'ensembles dans le cas du *grand univers* rejoignant ainsi les problématiques de la littérature (Boyen, CCS 2004 et Blanton et Aliasgari, IEEE Transactions on Information Forensics and Security 2013). Nous avons alors proposé une nouvelle direction pour traiter la réutilisabilité des Fuzzy Extractors puisque notre framework permet de convertir tout Fuzzy Extractor classique en un Fuzzy Extractor réutilisable via l'usage d'isométrie pseudo-entropiques ou *pseudoentropic isometry*. Également avec une sécurité basée sur l'oracle aléatoire, nous avons instancié ce framework pour la différence d'ensembles dans le cas du grand univers, refermant ainsi le problème de la réutilisabilité des Fuzzy Extractors basés sur cette métrique. Notre construction constitue également le premier Fuzzy Extractor à être réutilisable tout en corrigeant un taux d'erreur linéaire.

D'autre part et en prévision de l'avènement de l'Internet des Objets, nous avons ouvert la voie à une possible extension de la solution TA au cas multi-appareils par l'usage de schémas de signature de groupe. En plus d'avoir été le premier tel schéma post-quantique à être dynamique, la construction proposée au Chapitre 5 constituait la première signature de groupe basée sur la métrique de Hamming. Les tailles de clés et de signatures d'ordre $N^{1/\sqrt{log(N)}}$, qui bien que non logarithmiques, sont plus que satisfaisantes et meilleures que son unique concurrent dont les complexités sont linéaires pour un groupe statique (Ezerman *et al.*, Asiacrypt 2015).

Par une méthodologie proche, nous avons proposé une signature de groupe basée sur la métrique rang, jouissant de tailles de clés et de signatures logarithmiques en le nombre d'utilisateurs et dont l'instanciation surpasse tous ses concurrents post-quantiques. Bien que dynamique et jouissant de bons paramètres, ce schéma ne permet pas un utilisateur

donné d'intervenir dans le choix de sa clé secrète, allant ainsi à l'encontre de la philosophie du module FAST.

Perspectives

Nous avons vu que certaines limitations pratiques, notamment sur la connaissance des attributs nouvellement considérés, n'ont pas encore été résolues (limitations L4 et L5). L'objectif, à court terme, est d'étudier la population d'utilisateurs de l'application App pour permettre d'établir des statistiques à grande échelle. Ces statistiques nous permettront alors de revoir les estimations d'entropie proposées en 3.6 pour mener à une industrialisation du module FAST. D'autre part, cette industrialisation ne pourra se faire sans une optimisation d'implémentation puisque nos premiers résultats mènent à des temps de calcul de l'ordre du double du seuil imparti (Figure 3.11).

À plus long terme, la problématique d'authentification multi-appareils n'est que partiellement résolue et devra faire l'objet de nouvelles recherches.

Récapitulatif

Nous proposons par la Figure 6.8 de récapituler toutes les limitations que le module FAST aura résolues, soulevées puis résolues, partiellement résolues ou laissées comme travaux futurs.

Limitation	Définition	État
L0	Remplir l'objectif O1 sans interaction explicite avec l'utilisateur.	Résolue. À poursuivre.
L1	Identifier des attributs plus diversités (renforcer R7).	Résolue.
L2	Gestion des attributs variables.	Résolue.
L3	Gestion du type Fatal/Warning.	Résolue.
L4	Méconnaissance des attributs nouvellement identifiés.	Non résolue.
L5	Optimisation des temps de calculs.	Non résolue.
L6	Conception d'un FE réutilisable basé sur la différence d'ensembles dans le contexte du *grand univers*.	Résolue.
L7	Gestion des attributs qui varient au cours du temps.	Résolue.
L8	Gestion de la non-indépendance des attributs.	Résolue.
L9	Gestion du scénario multi-appareils avec anonymat de l'appareil impliqué.	Partiellement résolue.

Figure 6.8: Avancement de la résolution des limitations du projet TA

Bibliography

[1] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway, "A concrete security treatment of symmetric encryption," in *Foundations of Computer Science, 1997. Proceedings., 38th Annual Symposium on.* IEEE, 1997, pp. 394–403.

[2] S. Cauchie, "Method and system for authentication," Aug. 28 2013, eP Patent App. EP20,110,787,705.

[3] Q. Alamélou and S. Cauchie, "Assessment of a confidence level in the collection by a communication terminal of information relative to fingerprints," Apr. 27 2016, eP Patent App. EP20,140,306,675.

[4] Q. Alamélou, S. Cauchie, P. Gaborit, and O. Maas, "Authentification Numérique bassée sur les Fuzzy Extractors," in *Atelier sur la Protection pour la Vie Privée*, Cabourg, France, Jun. 2014.

[5] Q. Alamélou, P.-E. Berthier, C. Cachet, S. Cauchie, B. Fuller, and P. Gaborit, "Pseudoentropic isometries: A new framework for fuzzy extractor reusability," Cryptology ePrint Archive, Report 2016/1100, 2016.

[6] Q. Alamélou, O. Blazy, S. Cauchie, and P. Gaborit, "A Code-Based Group Signature Scheme," in *The 9th International Workshop on Coding and Cryptography 2015 WCC2015*, ser. Proceedings of the 9th International Workshop on Coding and Cryptography 2015 WCC2015, J.-P. T. Pascale Charpin, Nicolas Sendrier, Ed., Paris, France, Apr. 2015.

[7] ——, "A Practical Group Signature Scheme based on Rank Metric," in *Waifi 2016*, Ghent, Belgium, Jul. 2016.

[8] ——, "A code-based group signature scheme," *Designs, Codes and Cryptography*, vol. 82, no. 1, pp. 469–493, 2017.

[9] A. Yger and J.-A. Weil, *Mathématiques L3-Mathématiques appliquées: Cours complet avec 500 tests et exercices corrigés.* Pearson Education France, 2009.

[10] D. R.-C. R.C. Bose, "On a class of error correcting binary group codes," *Information and Control*, vol. 3, no. 1, pp. 68 – 79, 1960.

[11] A. Hocquenghem, "Codes correcteurs d'erreurs," *Chiffres*, vol. 2, no. 147-156, pp. 8–5, 1959.

[12] Y. Dodis, L. Reyzin, and A. Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," in *Advances in Cryptology - EUROCRYPT 2004*, ser. Lecture Notes in Computer Science, C. Cachin and J. Camenisch, Eds. Springer Berlin Heidelberg, 2004, vol. 3027, pp. 523–540.

[13] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," *SIAM J. Comput.*, vol. 38, no. 1, pp. 97–139, Mar. 2008.

[14] E. Berlekamp and L. Welch, "Error correction of algebraic block codes," Patent 4 633 470.

[15] A. Juels and M. Sudan, "A fuzzy vault scheme," *Des. Codes Cryptography*, vol. 38, no. 2, pp. 237–257, Feb. 2006.

[16] J. Faugère, V. Gauthier-Umaña, A. Otmani, L. Perret, and J. Tillich, "A distinguisher for high rate mceliece cryptosystems," in *2011 IEEE Information Theory Workshop, ITW 2011, Paraty, Brazil, October 16-20, 2011*, 2011, pp. 282–286.

[17] V. D. Goppa, "A new class of linear correcting codes," *Problemy Peredachi Informatsii*, vol. 6, no. 3, pp. 24–30, 1970.

[18] E. M. Gabidulin, "Theory of codes with maximum rank distance," *Probl. Peredachi Inf., Volume 21, Issue 1, pages 3-16*, 1985.

[19] P. Loidreau, "Properties of codes in rank metric," *CoRR*, vol. abs/cs/0610057, 2006.

[20] P. Gaborit, G. Murat, O. Ruatta, and G. Zémor, "Low Rank Parity Check codes and their application to cryptography," in *WCC 13*, Bergen, Norway, Apr. 2013.

[21] A. Kerckhoffs, "La cryptographie militaire," *Journal des sciences militaires*, pp. 5–83, January 1883.

[22] S. Goldwasser and S. Micali, "Probabilistic encryption," *Journal of Computer and System Sciences*, vol. 28, no. 2, pp. 270 – 299, 1984.

[23] M. Bellare and P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols," in *Proceedings of the 1st ACM Conference on Computer and Communications Security*, ser. CCS '93. New York, NY, USA: ACM, 1993, pp. 62–73.

[24] S. Goldwasser and Y. T. Kalai, "On the (in) security of the fiat-shamir paradigm," in *Foundations of Computer Science, 2003. Proceedings. 44th Annual IEEE Symposium on.* IEEE, 2003, pp. 102–113.

[25] R. Canetti, O. Goldreich, and S. Halevi, "The random oracle methodology, revisited," *J. ACM*, vol. 51, no. 4, pp. 557–594, Jul. 2004.

[26] N. Koblitz and A. J. Menezes, "The random oracle model: a twenty-year retrospective," *Designs, Codes and Cryptography*, vol. 77, no. 2, pp. 587–610, 2015.

[27] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, Feb. 1978.

[28] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," in *Workshop on the Theory and Application of Cryptographic Techniques*. Springer, 1984, pp. 10–18.

[29] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM review*, vol. 41, no. 2, pp. 303–332, 1999.

[30] R. G. Michael and S. J. David, "Computers and intractability: a guide to the theory of np-completeness," *WH Free. Co., San Fr*, 1979.

[31] E. Bresson, "Protocoles cryptographiques pour l'authentification et l'anonymat dans les groupes," Ph.D. dissertation, École polytechnique, 2002.

[32] O. Goldreich, *Foundations of cryptography: volume 1, Basic Tools*. Cambridge University Press, 2001.

[33] O. Goldreich, S. Goldwasser, and S. Micali, "How to construct random functions," *Journal of the ACM (JACM)*, vol. 33, no. 4, pp. 792–807, 1986.

[34] N. Nisan and D. Zuckerman, "Randomness is linear in space," *Journal of Computer and System Sciences*, vol. 52, no. 1, pp. 43–52, 1996.

[35] L. Trevisan, "Extractors and pseudorandom generators," *Journal of the ACM*, vol. 48, no. 4, pp. 860–879, 2001.

[36] R. Shaltiel, "Recent developments in explicit constructions of extractors," *Bulletin of the EATCS*, vol. 77, no. 67-95, p. 10, 2002.

[37] H. Krawczyk, "Cryptographic extraction and key derivation: The hkdf scheme," in *Annual Cryptology Conference*. Springer, 2010, pp. 631–648.

[38] D. Dachman-Soled, R. Gennaro, H. Krawczyk, and T. Malkin, "Computational extractors and pseudorandomness," in *Theory of Cryptography Conference*. Springer, 2012, pp. 383–403.

[39] H. Krawczyk, R. Canetti, and M. Bellare, "Hmac: Keyed-hashing for message authentication," 1997.

[40] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Trans. Inf. Theor.*, vol. 22, no. 6, pp. 644–654, Sep. 2006.

[41] "Recommendation forthe Triple Data Encryption Algorithm (TDEA) Block Cipher," http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-67r1.pdf.

[42] J. Daemen and V. Rijmen, *The design of Rijndael: AES-the advanced encryption standard*. Springer Science & Business Media, 2013.

[43] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway, *Relations among notions of security for public-key encryption schemes*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 26–45.

[44] M. Dworkin, "Nist special publication 800-38b recommendation for block cipher modes of operation: The cmac mode for authentication," 2005.

[45] D. McGrew and J. Viega, "The galois/counter mode of operation (gcm)," *Submission to NIST Modes of Operation Process*, vol. 20, 2004.

[46] J. Song, R. Poovendran, J. Lee, and T. Iwata, "The aes-cmac algorithm," Tech. Rep., 2006.

[47] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, 1986, pp. 186–194.

[48] S. Goldwasser, S. Micali, and R. L. Rivest, "A digital signature scheme secure against adaptive chosen-message attacks," *SIAM J. Comput.*, vol. 17, no. 2, pp. 281–308, Apr. 1988.

[49] D. Chaum and E. van Heyst, "Group signatures," in *Advances in Cryptology — EUROCRYPT '91*, ser. Lecture Notes in Computer Science, D. Davies, Ed. Springer Berlin Heidelberg, 1991, vol. 547, pp. 257–265.

[50] G. Brassard, D. Chaum, and C. Crépeau, "Minimum disclosure proofs of knowledge," *J. Comput. Syst. Sci.*, vol. 37, no. 2, pp. 156–189, Oct. 1988.

[51] M. O. Rabin, *Digital signatures*, R. J. Lipton, D. P. Dobkin, and A. K. Jones, Eds. Orlando, FL, USA: Academic Press, Inc., 1978.

[52] U. Fiege, A. Fiat, and A. Shamir, "Zero knowledge proofs of identity," in *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, ser. STOC '87. New York, NY, USA: ACM, 1987, pp. 210–217.

[53] U. Feige and A. Shamir, *Zero Knowledge Proofs of Knowledge in Two Rounds.* New York, NY: Springer New York, 1990, pp. 526–544.

[54] ——, "Witness indistinguishable and witness hiding protocols," in *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, 1990, pp. 416–426.

[55] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof-systems," in *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*, ser. STOC '85. New York, NY, USA: ACM, 1985, pp. 291–304.

[56] T. Itoh and K. Sakurai, "On the complexity of constant round zkip of possession of knowledge," in *Proceedings of the International Conference on the Theory and Applications of Cryptology: Advances in Cryptology*, ser. ASIACRYPT '91. London, UK, UK: Springer-Verlag, 1993, pp. 331–345.

[57] E. R. Berlekamp, R. J. McEliece, and H. C. A. van Tilborg, "On the inherent intractability of certain coding problems (corresp.)," *IEEE Trans. Information Theory*, vol. 24, no. 3, pp. 384–386, 1978.

[58] N. Courtois, M. Finiasz, and N. Sendrier, "How to achieve a McEliece-based digital signature scheme," in *Advances in Cryptology - ASIACRYPT 2001*, ser. LNCS, C. Boyd, Ed., vol. 2248. Springer, 2001, pp. 157–174.

[59] A. Vardy, "The intractability of computing the minimum distance of a code," *Information Theory, IEEE Transactions on*, vol. 43, no. 6, pp. 1757–1766, Nov 1997.

[60] R. McEliece, "A public-key cryptosystem based on algebraic," *Coding Thv*, vol. 4244, pp. 114–116, 1978.

[61] A. Canteaut and N. Sendrier, *Cryptanalysis of the Original McEliece Cryptosystem.* Springer Berlin Heidelberg, 1998.

[62] D. J. Bernstein, T. Lange, and C. Peters, *Attacking and Defending the McEliece Cryptosystem.* Springer Berlin Heidelberg, 2008.

[63] M. Finiasz and N. Sendrier, "Security bounds for the design of code-based cryptosystems," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2009, pp. 88–105.

[64] D. J. Bernstein, T. Lange, and C. Peters, "Smaller decoding exponents: ball-collision decoding," in *Annual Cryptology Conference*. Springer, 2011, pp. 743–760.

[65] H. Niederreiter, "Knapsack-type cryptosystems and algebraic coding theory," *Problems of Control and Information Theory*, vol. 15, no. 2, pp. 159–166, 1986.

[66] Y. X. Li, R. H. Deng, and X. M. Wang, "On the equivalence of mceliece's and niederreiter's public-key cryptosystems," *IEEE Transactions on Information Theory*, vol. 40, no. 1, pp. 271–273, 1994.

[67] J. Stern, "A new paradigm for public key identification," *IEEE Trans. Information Theory*, vol. 42, no. 6, pp. 1757–1768, 1996.

[68] P. S. L. M. Barreto, P.-L. Cayrel, R. Misoczki, and R. Niebuhr, *Quasi-Dyadic CFS Signatures*. Springer Berlin Heidelberg, 2011.

[69] M. Finiasz, *Parallel-CFS*. Springer Berlin Heidelberg, 2011.

[70] D. Wagner, "A generalized birthday problem," in *Annual International Cryptology Conference*. Springer, 2002, pp. 288–304.

[71] C. H. Bennett, G. Brassard, and J.-M. Robert, "Privacy amplification by public discussion," *SIAM Journal on Computing*, vol. 17, no. 2, pp. 210–229, 1988.

[72] R. Pappu, B. Recht, J. Taylor, and N. Gershenfeld, "Physical one-way functions," *Science*, vol. 297, no. 5589, pp. 2026–2030, 2002.

[73] "Biometrics on the smartphone: The future of mobile authentication," http://whatsnext.nuance.com/customer-experience/biometrics-smartphone-future-mobile-authentication/.

[74] "Mobile Biometric Authentication. White Paper." http://www.biometricupdate.com/wp-content/uploads/2014/08/Mobile-Biometric-Authentication-Report.pdf.

[75] "Notre sélection des meilleurs smartphones avec lecteur d'empreintes digitales," http://www.phonandroid.com/selection-meilleurs-smartphones-lecteur-empreintes-digitales.html.

[76] "Zoom sur le capteur d'iris du samsung galaxy note 7," http://www.frandroid.com/marques/samsung/381623_zoom-capteur-diris-samsung-galaxy-note-7.

[77] "La reconnaissance de l'iris déverrouillera 300 millions de smartphones en 2021," http://www.usine-digitale.fr/article/la-reconnaissance-de-l-iris-deverrouillera-300-millions-de-smartphones-en-2021.N44344.

[78] A. Juels and M. Wattenberg, "A fuzzy commitment scheme," in *Proceedings of the 6th ACM Conference on Computer and Communications Security*, ser. CCS '99. New York, NY, USA: ACM, 1999, pp. 28–36.

[79] G. I. Davida, Y. Frankel, and B. J. Matt, "On enabling secure applications through off-line biometric identification," in *Proceedings. 1998 IEEE Symposium on Security and Privacy (Cat. No.98CB36186)*, May 1998, pp. 148–157.

[80] G. I. Davida, Y. Frankel, B. J. Matt, and R. Peralta, "On the relation of error correction and cryptography to an off line biometric based identification scheme," 1999.

[81] J. Bringer, H. Chabanne, G. Cohen, B. Kindarji, and G. Zemor, "Optimal iris fuzzy sketches," in *2007 First IEEE International Conference on Biometrics: Theory, Applications, and Systems*, Sept 2007, pp. 1–6.

[82] J.-P. Linnartz and P. Tuyls, *New Shielding Functions to Enhance Privacy and Prevent Misuse of Biometric Templates.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 393–402.

[83] B. Fuller, X. Meng, and L. Reyzin, *Computational Fuzzy Extractors.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 174–193.

[84] G. Cohen and G. Zemor, "The wiretap channel applied to biometrics," in *ISITA*, Parma, Italy, 2004, pp. 1–5. [Online]. Available: https://hal.archives-ouvertes.fr/hal-00359822

[85] X. Boyen, "Reusable cryptographic fuzzy extractors," in *Proceedings of the 11th ACM Conference on Computer and Communications Security*, ser. CCS '04. New York, NY, USA: ACM, 2004, pp. 82–91.

[86] X. Boyen, Y. Dodis, J. Katz, R. Ostrovsky, and A. Smith, *Secure Remote Authentication Using Biometric Data.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 147–163.

[87] Y. Dodis, J. Katz, L. Reyzin, and A. Smith, *Robust Fuzzy Extractors and Authenticated Key Agreement from Close Secrets.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 232–250.

[88] R. Cramer, Y. Dodis, S. Fehr, C. Padró, and D. Wichs, *Detection of Algebraic Manipulation with Applications to Robust Secret Sharing and Fuzzy Extractors*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 471–488.

[89] Y. Dodis, B. Kanukurthi, J. Katz, L. Reyzin, and A. D. Smith, "Robust fuzzy extractors and authenticated key agreement from close secrets," *IEEE Trans. Information Theory*, vol. 58, no. 9, pp. 6207–6222, 2012.

[90] K. Simoens, P. Tuyls, and B. Preneel, "Privacy weaknesses in biometric sketches," in *2009 30th IEEE Symposium on Security and Privacy*, 2009, pp. 188–203.

[91] M. Blanton and M. Aliasgari, "On the (non-)reusability of fuzzy sketches and extractors and security improvements in the computational setting," *IACR Cryptology ePrint Archive*, vol. 2012, p. 608, 2012.

[92] ——, "Analysis of reusability of secure sketches and fuzzy extractors," *IEEE Trans. Information Forensics and Security*, vol. 8, no. 9, pp. 1433–1445, 2013.

[93] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," *Journal of the ACM (JACM)*, vol. 56, no. 6, p. 34, 2009.

[94] R. Canetti, B. Fuller, O. Paneth, L. Reyzin, and A. Smith, *Advances in Cryptology – EUROCRYPT 2016: 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, ch. Reusable Fuzzy Extractors for Low-Entropy Distributions, pp. 117–146.

[95] I. Buhan, J. Doumen, P. H. Hartel, and R. N. J. Veldhuis, "Fuzzy extractors for continuous distributions," in *Proceedings of the 2007 ACM Symposium on Information, Computer and Communications Security, ASIACCS 2007, Singapore, March 20-22, 2007*, 2007, pp. 353–355.

[96] V. P. Parente and J. van de Graaf, *A Practical Fuzzy Extractor for Continuous Features*. Cham: Springer International Publishing, 2016, pp. 241–258.

[97] F. Monrose, M. K. Reiter, Q. Li, and S. Wetzel, "Cryptographic key generation from voice," in *Proceedings of the 2001 IEEE Symposium on Security and Privacy*, ser. SP '01. Washington, DC, USA: IEEE Computer Society, 2001, pp. 202–.

[98] Y.-J. Chang, W. Zhang, and T. Chen, "Biometrics-based cryptographic key generation," in *2004 IEEE International Conference on Multimedia and Expo (ICME) (IEEE Cat. No.04TH8763)*, vol. 3, June 2004, pp. 2203–2206 Vol.3.

[99] W. Zhang, Y.-J. Chang, and T. Chen, "Optimal thresholding for key generation based on biometrics," in *Image Processing, 2004. ICIP'04. 2004 International Conference on*, vol. 5. IEEE, 2004, pp. 3451–3454.

[100] A. Van Herrewege, S. Katzenbeisser, R. Maes, R. Peeters, A.-R. Sadeghi, I. Verbauwhede, and C. Wachsmann, *Reverse Fuzzy Extractors: Enabling Lightweight Mutual Authentication for PUF-Enabled RFIDs*. Springer Berlin Heidelberg, 2012.

[101] "GlobalPlatform made simple guide: Secure Element," https://www.globalplatform.org/mediaguideSE.asp.

[102] "Host-based Card Emulation," https://developer.android.com/guide/topics/connectivity/nfc/hce.html.

[103] "EMVCo," https://www.emvco.com/specifications.aspx?id=223.

[104] "Visa Ready Program," https://technologypartner.visa.com/VisaReady/MobilePayments.aspx#CloudBasedPayments.

[105] "Mastercard OnLine. Documentation," http://mastercard-mobilepartner.com/documentation.html#2.

[106] "Référentiels d'Exigences," https://www.ssi.gouv.fr/administration/qualifications/prestataires-de-services-de-confiance-qualifies/referentiels-exigences/.

[107] "Authentification. règles et recommandations concernant les mécanismes d'authentification de niveau de robustesse standard," https://www.ssi.gouv.fr/archive/fr/politique_produit/catalogue/pdf/authentification_robustesse_standard_v0-13.pdf.

[108] "National Institute of Standards and Technology," http://csrc.nist.gov/groups/ST/toolkit/.

[109] "Recommandations cnil," https://www.cnil.fr/professionnel.

[110] P. Eckersley, "How unique is your web browser?" in *Privacy Enhancing Technologies, 10th International Symposium, PETS 2010, Berlin, Germany, July 21-23, 2010. Proceedings*, 2010, pp. 1–18.

[111] A. Kurtz, H. Gascon, T. Becker, K. Rieck, and F. Freiling, "Fingerprinting mobile devices using personalized configurations," *Proceedings on Privacy Enhancing Technologies*, vol. 2016, no. 1, pp. 4–19, 2016.

[112] “Bluekrypt. cryptographic key length recommendation.” https://www.keylength.com/fr/5/.

[113] “Panopticlick. Is your browser safe against tracking?” https://panopticlick.eff.org/.

[114] “Am I Unique?” https://amiunique.org/.

[115] “Fingerprintjs2,” https://github.com/Valve/fingerprintjs2.

[116] “FIDO Alliance,” https://fidoalliance.org/?s=passwor.

[117] “FIDO Alliance,” https://fidoalliance.org/fido-alliance-opens-worldwide-cooperation-and-liaison-

[118] M. E. Khan and F. Khan, “A comparative study of white box, black box and grey box testing techniques,” *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 3, no. 6, 2012.

[119] P. C. Kocher, *Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems*. Springer Berlin Heidelberg, 1996.

[120] S. Chow, P. Eisen, H. Johnson, and P. C. Van Oorschot, *White-Box Cryptography and an AES Implementation*. Springer Berlin Heidelberg, 2003.

[121] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan, and K. Yang, *On the (Im)possibility of Obfuscating Programs*, 2001.

[122] ——, “On the (im)possibility of obfuscating programs,” *J. ACM*, vol. 59, no. 2, pp. 6:1–6:48, May 2012.

[123] Z. Brakerski and O. Dagmi, “Shorter circuit obfuscation in challenging security models,” in *Security and Cryptography for Networks - 10th International Conference, SCN 2016, Amalfi, Italy, August 31 - September 2, 2016, Proceedings*, 2016, pp. 551–570.

[124] Y. D. Mulder, “White-box cryptography. analysis of white-box aes implementations.” Ph.D. dissertation, KU LEUVEN, 2014.

[125] T. Lepoint, M. Rivain, Y. D. Mulder, P. Roelse, and B. Preneel, “Two attacks on a white-box AES implementation,” in *Selected Areas in Cryptography - SAC 2013 - 20th International Conference, Burnaby, BC, Canada, August 14-16, 2013, Revised Selected Papers*, 2013, pp. 265–285.

[126] “Device Fingerprinting and Fraud Protection ,” Device-Fingerprinting-and-Online-Fraud-Protection-Whitepaper.pdf.

[127] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, and J. Schmidhuber, "Modeling attacks on physical unclonable functions," in *Proceedings of the 17th ACM Conference on Computer and Communications Security*, ser. CCS '10. New York, NY, USA: ACM, 2010, pp. 237–249. [Online]. Available: http://doi.acm.org/10.1145/1866307.1866335

[128] U. Ruhrmair and M. van Dijk, "Pufs in security protocols: Attack models and security evaluations," *2012 IEEE Symposium on Security and Privacy*, pp. 286–300, 2013.

[129] "GGlobalPlatform made simple guide: Trusted Execution Environment (TEE) Guide," http://www.globalplatform.org/mediaguidetee.asp#_Toc419214135.

[130] K. Boda, A. M. Földes, G. G. Gulyás, and S. Imre, "User tracking on the web via cross-browser fingerprinting," in *Proceedings of the 16th Nordic Conference on Information Security Technology for Applications*, ser. NordSec'11. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 31–46.

[131] K. Mowery and H. Shacham, "Pixel perfect: Fingerprinting canvas in HTML5," in *Proceedings of W2SP 2012*, M. Fredrikson, Ed. IEEE Computer Society, May 2012.

[132] N. Nikiforakis, A. Kapravelos, W. Joosen, C. Kruegel, F. Piessens, and G. Vigna, "Cookieless monster: Exploring the ecosystem of web-based device fingerprinting," in *Proceedings of the 2013 IEEE Symposium on Security and Privacy*, ser. SP '13. Washington, DC, USA: IEEE Computer Society, 2013, pp. 541–555.

[133] G. Acar, C. Eubank, S. Englehardt, M. Juarez, A. Narayanan, and C. Diaz, "The web never forgets: Persistent tracking mechanisms in the wild," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '14. New York, NY, USA: ACM, 2014, pp. 674–689.

[134] S. Kamkar, "Evercookie-virtually irrevocable persistent cookies," *His Blog*, vol. 9, 2010.

[135] A. Soltani, S. Canty, Q. Mayo, L. Thomas, and C. J. Hoofnagle, "Flash cookies and privacy." in *AAAI spring symposium: intelligent information privacy management*, vol. 2010, 2010, pp. 158–163.

[136] B. Krishnamurthy and C. Wills, "Privacy diffusion on the web: a longitudinal perspective," in *Proceedings of the 18th international conference on World wide web*. ACM, 2009, pp. 541–550.

[137] J. R. Mayer and J. C. Mitchell, "Third-party web tracking: Policy and technology," in *2012 IEEE Symposium on Security and Privacy*. IEEE, 2012, pp. 413–427.

[138] J. Lukas, J. Fridrich, and M. Goljan, "Digital camera identification from sensor pattern noise," *IEEE Transactions on Information Forensics and Security*, vol. 1, no. 2, pp. 205–214, 2006.

[139] M. Chen, J. Fridrich, M. Goljan, and J. Lukás, "Determining image origin and integrity using sensor noise," *IEEE Transactions on Information Forensics and Security*, vol. 3, no. 1, pp. 74–90, 2008.

[140] H. Bojinov, Y. Michalevsky, G. Nakibly, and D. Boneh, "Mobile device identification via sensor fingerprinting," *arXiv preprint arXiv:1408.1416*, 2014.

[141] A. Das, N. Borisov, and M. Caesar, "Do you hear what i hear?: Fingerprinting smart devices through embedded acoustic components," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '14. New York, NY, USA: ACM, 2014, pp. 441–452.

[142] Z. Zhou, W. Diao, X. Liu, and K. Zhang, "Acoustic fingerprinting revisited: Generate stable device id stealthily with inaudible sound," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2014, pp. 429–440.

[143] S. Dey, N. Roy, W. Xu, R. R. Choudhury, and S. Nelakuditi, "Accelprint: Imperfections of accelerometers make smartphones trackable." in *NDSS*, 2014.

[144] S. Seneviratne, A. Seneviratne, P. Mohapatra, and A. Mahanti, "Predicting user traits from a snapshot of apps installed on a smartphone," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 18, no. 2, pp. 1–8, Jun. 2014.

[145] J. P. Achara, G. Acs, and C. Castelluccia, "On the unicity of smartphone applications," in *Proceedings of the 14th ACM Workshop on Privacy in the Electronic Society*, ser. WPES '15. New York, NY, USA: ACM, 2015, pp. 27–36.

[146] A. K. Jain, K. Nandakumar, and A. Ross, "50 years of biometric research: Accomplishments, challenges, and opportunities," *Pattern Recognition Letters*, 2016.

[147] B. Barak, "Lecture 4 - Computational Indistinguishability, Pseudorandom Generators." https://www.cs.princeton.edu/courses/archive/fall07/cos433/lec4.pdf.

[148] "Most Common First Names and Last Names in the U.S." http://names.mongabay.com/male_names.htm.

[149] "Rooter Security. Choosing an SSID." http://routersecurity.org/SSID.php.

[150] "Recommended settings for Wi-Fi routers and access points." https://support.apple.com/en-us/HT202068.

[151] "4 GSM 3GPP Specifications and Rulesets." https://docs.oracle.com/cd/E36032_01/doc.722/e36045/gsm_mobile_cartridge.htm.

[152] "International Mobile Subscriber Identity." https://fr.wikipedia.org/wiki/International_Mobile_Subscriber_Identity.

[153] J. Daugman, "How iris recognition works," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, pp. 21–30, 2002.

[154] K. Nandakumar, A. K. Jain, and S. Pankanti, "Fingerprint-based fuzzy vault: Implementation and performance," *IEEE Transactions on Information Forensics and Security*, vol. 2, no. 4, pp. 744–757, 2007.

[155] A. K. Jain, K. Nandakumar, and A. Nagar, "Biometric template security," *EURASIP J. Adv. Signal Process*, vol. 2008, pp. 113:1–113:17, Jan. 2008.

[156] U. Uludag, S. Pankanti, S. Prabhakar, and A. K. Jain, "Biometric cryptosystems: issues and challenges," *Proceedings of the IEEE*, vol. 92, no. 6, pp. 948–960, 2004.

[157] U. Uludag, A. Ross, and A. Jain, "Biometric template selection and update: a case study in fingerprints," *Pattern Recognition*, vol. 37, no. 7, pp. 1533–1542, 2004.

[158] J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby, "A pseudorandom generator from any one-way function," *SIAM Journal on Computing*, vol. 28, no. 4, pp. 1364–1396, 1999.

[159] C.-Y. Hsiao, C.-J. Lu, and L. Reyzin, "Conditional computational entropy, or toward separating pseudoentropy from compressibility," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2007, pp. 169–186.

[160] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway, "A concrete security treatment of symmetric encryption," in *Proceedings of the 38th Annual Symposium on Foundations of Computer Science*, ser. FOCS '97. Washington, DC, USA: IEEE Computer Society, 1997, pp. 394–.

[161] M. Bellare, H. Shi, and C. Zhang, "Foundations of group signatures: The case of dynamic groups," in *Topics in Cryptology – CT-RSA 2005*, ser. Lecture Notes in Computer Science, A. Menezes, Ed. Springer Berlin Heidelberg, 2005, vol. 3376, pp. 136–153.

[162] J. Stern, "A new identification scheme based on syndrome decoding," in *Advances in Cryptology — CRYPTO' 93*, ser. Lecture Notes in Computer Science, D. Stinson, Ed. Springer Berlin Heidelberg, 1994, vol. 773, pp. 13–21.

[163] M. Bellare, D. Micciancio, and B. Warinschi, "Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions," in *Advances in Cryptology — EUROCRYPT 2003*, ser. Lecture Notes in Computer Science, E. Biham, Ed. Springer Berlin Heidelberg, 2003, vol. 2656, pp. 614–629.

[164] D. Boneh, X. Boyen, and H. Shacham, "Short group signatures," in *Advances in Cryptology – CRYPTO 2004*, ser. Lecture Notes in Computer Science, M. Franklin, Ed. Springer Berlin Heidelberg, 2004, vol. 3152, pp. 41–55.

[165] D. Boneh and H. Shacham, "Group signatures with verifier-local revocation," in *Proceedings of CCS 2004*. ACM Press, 2004, pp. 168–177.

[166] J. Camenisch and A. Lysyanskaya, "Signature schemes and anonymous credentials from bilinear maps," in *Advances in Cryptology - CRYPTO 2004, 24th Annual International CryptologyConference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, 2004, pp. 56–72.

[167] C. Delerablée and D. Pointcheval, "Dynamic fully anonymous short group signatures," in *Progressin Cryptology - VIETCRYPT 2006, First International Conference on Cryptology in Vietnam, Hanoi, Vietnam, September 25-28, 2006, Revised Selected Papers*, 2006, pp. 193–210.

[168] J. Groth, "Fully anonymous group signatures without random oracles," in *Advances in Cryptology - ASIACRYPT 2007, 13th International Conference on the Theory and Application of Cryptology and Information Security, Kuching, Malaysia, December 2-6, 2007, Proceedings*, 2007, pp. 164–180.

[169] A. Kiayias, Y. Tsiounis, and M. Yung, "Traceable signatures," in *Advances in Cryptology - EUROCRYPT 2004*, ser. Lecture Notes in Computer Science, C. Cachin and J. Camenisch, Eds. Springer Berlin Heidelberg, 2004, vol. 3027, pp. 571–589.

[170] B. Libert and M. Yung, "Efficient traceable signatures in the standard model," in *Pairing-Based Cryptography – Pairing 2009*, ser. Lecture Notes in Computer Science, H. Shacham and B. Waters, Eds. Springer Berlin Heidelberg, 2009, vol. 5671, pp. 187–205.

[171] S. Gordon, J. Katz, and V. Vaikuntanathan, "A group signature scheme from lattice assumptions," in *Advances in Cryptology - ASIACRYPT 2010*, ser. Lecture Notes in Computer Science, M. Abe, Ed. Springer Berlin Heidelberg, 2010, vol. 6477, pp. 395–412.

[172] F. Laguillaumie, A. Langlois, B. Libert, and D. Stehlé, "Lattice-based group signatures with logarithmic signature size," in *Advances in Cryptology - ASIACRYPT 2013*, ser. Lecture Notes in Computer Science, K. Sako and P. Sarkar, Eds. Springer Berlin Heidelberg, 2013, vol. 8270, pp. 41–61.

[173] A. Langlois, S. Ling, K. Nguyen, and H. Wang, "Lattice-based group signature scheme with verifier-local revocation," in *PKC 2014*, ser. Lecture Notes in Computer Science, H. Krawczyk, Ed. Springer Berlin Heidelberg, 2014, vol. 8383, pp. 345–361.

[174] S. Ling, K. Nguyen, and H. Wang, "Group signatures from lattices: Simpler, tighter, shorter, ring-based," in *PKC 2015*, ser. Lecture Notes in Computer Science, J. Katz, Ed. Springer Berlin Heidelberg, 2015, vol. 9020, pp. 427–449.

[175] P. Q. Nguyen, J. Zhang, and Z. Zhang, "Simpler efficient group signatures from lattices," in *Public-Key Cryptography - PKC 2015 - 18th IACR International Conference on Practice and Theory in Public-Key Cryptography, Gaithersburg, MD, USA, March 30 - April 1, 2015, Proceedings*, 2015, pp. 401–426.

[176] M. F. Ezerman, H. T. Lee, S. Ling, K. Nguyen, and H. Wang, "A provably secure group signature scheme from code-based assumptions," in *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part I*, 2015, pp. 260–285.

[177] B. Libert, F. Mouhartem, and K. Nguyen, "A lattice-based group signature scheme with message-dependent opening," in *Applied Cryptography and Network Security - 14th International Conference, ACNS 2016, Guildford, UK, June 19-22, 2016. Proceedings*, 2016, pp. 137–155.

[178] S. Arora, L. Babai, J. Stern, and Z. Sweedyk, "The hardness of approximate optimia in lattices, codes, and systems of linear equations," in *34th Annual Symposium on*

Foundations of Computer Science, Palo Alto, California, USA, 3-5 November 1993, 1993, pp. 724–733.

[179] M. Bellare, C. Namprempre, D. Pointcheval, and M. Semanko, "The one-more-rsa-inversion problems and the security of chaum's blind signature scheme," *J. Cryptology*, vol. 16, no. 3, pp. 185–215, 2003.

[180] K. P. Mathew, S. Vasant, and C. P. Rangan, *A Provably Secure Signature and Signcryption Scheme Using the Hardness Assumptions in Coding Theory.* Cham: Springer International Publishing, 2014, pp. 342–362.

[181] A. Becker, A. Joux, A. May, and A. Meurer, "Decoding random binary linear codes in 2 n/20: How 1 + 1 = 0 improves information set decoding," in *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, 2012, pp. 520–536.

[182] B. Libert, S. Ling, F. Mouhartem, K. Nguyen, and H. Wang, *Signature Schemes with Efficient Protocols and Dynamic Group Signatures from Lattice Assumptions.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 373–403. [Online]. Available: http://dx.doi.org/10.1007/978-3-662-53890-6_13

[183] P. Gaborit, O. Ruatta, J. Schrek, and G. Zémor, "Ranksign: An efficient signature algorithm based on the rank metric," in *Post-Quantum Cryptography*, ser. Lecture Notes in Computer Science, M. Mosca, Ed. Springer International Publishing, 2014, vol. 8772, pp. 88–107.

[184] P. Gaborit, O. Ruatta, J. Schrek, and G. Zémor, "Ranksign : an efficient signature algorithm based on the rank metric," *CoRR*, vol. abs/1606.00629, 2016.

[185] P. Gaborit, J. Schrek, and G. Zémor, "Full cryptanalysis of the chen identification protocol," in *Post-Quantum Cryptography*, ser. Lecture Notes in Computer Science, B.-Y. Yang, Ed. Springer Berlin Heidelberg, 2011, vol. 7071, pp. 35–50.

[186] J. Fischer and J. Stern, "An efficient pseudo-random generator provably as secure as syndrome decoding," in *Advances in Cryptology - EUROCRYPT '96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding*, 1996, pp. 245–255.

[187] B. Libert, S. Ling, K. Nguyen, and H. Wang, *Zero-Knowledge Arguments for Lattice-Based Accumulators: Logarithmic-Size Ring Signatures and Group Signatures Without Trapdoors.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2016.

[188] P. Gaborit, O. Ruatta, J. Schrek, and G. Zémor, "New results for rank-based cryptography," in *Progress in Cryptology - AFRICACRYPT 2014 - 7th International Conference on Cryptology in Africa, Marrakesh, Morocco, May 28-30, 2014. Proceedings*, 2014, pp. 1–12.

[189] P. Gaborit and G. Zémor, "On the hardness of the decoding and the minimum distance problems for rank codes," *IEEE Transactions on Information Theory*, vol. 62, no. 12, pp. 7245–7252, 2016.

[190] P. Gaborit, O. Ruatta, and J. Schrek, "On the complexity of the rank syndrome decoding problem," *Information Theory, IEEE Transactions on*, vol. 62, no. 2, pp. 1006–1019, Feb 2016.

[191] P. Gaborit, A. Hauteville, and J. Tillich, "Ranksynd a PRNG based on rank metric," in *Post-Quantum Cryptography - 7th International Workshop, PQCrypto 2016, Fukuoka, Japan, February 24-26, 2016, Proceedings*, 2016, pp. 18–28.

[192] K. Chen, "A new identification algorithm," in *Cryptography: Policy and Algorithms, International Conference, Brisbane, Queensland, Australia, July 3-5, 1995, Proceedings*, 1995, pp. 244–249.