UNIVERSITÉ DE LIMOGES
ÉCOLE DOCTORALE « Sciences et Ingénierie pour l'Information »
FACULTÉ DES SCIENCES ET TECHNIQUES
Laboratoire XLIM(DMI), équipe PICC

Thèse N°

Signature et identification pour l'anonymat basées sur les réseaux

THÈSE
pour obtenir le grade de
DOCTEUR DE L'UNIVERSITÉ DE LIMOGES
Discipline / Spécialité : Informatique
présentée et soutenue par
Slim BETTAIEB
le 26 septembre 2014

Thèse co-encadrée par Carlos AGUILAR MELCHOR et Philippe GABORIT

JURY

Rapporteurs :	
M. Gilles ZÉMOR	Professeur à l'Université de Bordeaux
M. Damien VERGNAUD	Maître de conférences HDR à l'École Normale Supérieure
Examinateurs :	
M. Thierry BERGER	Professeur à l'Université de Limoges
M. Philippe GABORIT	Professeur à l'Université de Limoges
M. Carlos AGUILAR MELCHOR	Maître de conférences HDR à l'Université de Toulouse
M. Marc RYBOWICZ	Maître de conférences à l'Université de Limoges

À ma famille.

Remerciments

Je tiens tout d'abord à remercier monsieur monsieur Gilles Zémor et monsieur Damien Vergnaud d'avoir accepté d'examiner mes travaux de recherche.

J'adresse mes remerciements les plus sincères à Philippe Gaborit et Carlos Aguilar Melchor pour avoir dirigé cette thèse et pour la confiance qu'ils ont su m'accorder.

Mes remerciements vont également à mes collègues, Julien, Suzy, Richard, Achref et tous les autres. Un grand merci à mes amis limogeauds : Laura, Martin, Fourti, Mourad, Lotfi, Irma, Jouini pour tous ces bons moments passés en leur compagnie.

Je remercie ma famille qui m'a toujours épaulé et encouragé, même dans les périodes les plus dures.

Résumé

La cryptographie basée sur les réseaux a connu depuis quelques années un très fort développement notamment du fait qu'il existe des systèmes cryptographiques basés sur les réseaux avec des propriétés de sécurité plus fortes que dans les cas plus classiques de théorie des nombres. Les problèmes difficiles des réseaux, par exemple le problème de trouver des vecteurs courts non nuls, semblent résister aux attaques utilisant des ordinateurs quantiques et les meilleurs algorithmes qui existent pour les résoudre sont exponentiels en fonction du temps. L'objet de cette thèse est la construction de primitives cryptographiques à clé publique pour l'anonymat dont la sécurité repose sur des problèmes difficiles des réseaux.

Nous nous intéressons aux schémas de signature de cercle. Tout d'abord, nous proposons une nouvelle définition d'anonymat et nous exposons un nouveau schéma de signature de cercle. Ensuite, nous donnons une étude de sécurité rigoureuse suivant deux définitions de résistance la contrefaçon. La première est la résistance à la contrefaçon contre les attaques à sous-cercles choisis et la deuxième est la résistance à la contrefaçon contre les attaques de corruption interne.

Nous présentons ensuite un nouveau schéma d'identification de cercle et nous développons une analyse complète de sa sécurité. Enfin, nous montrons que les techniques utilisées pour construire le schéma précédent peuvent être utilisées pour construire un schéma d'identification de cercle à seuil.

Mots-clés. Cryptographie à clé publique, réseaux euclidiens, signature de cercle, identification de cercle, identification de cercle à seuil, anonymat, sécurité prouvée.

Abstract

Lattice-based cryptography has known during the last decade rapid developments thanks to stronger security properties. In fact, there exist lattice-based cryptographic systems whose security is stronger than those based on the conventional number theory approach. The hard problems of lattices, for example the problem of finding short non-zero vectors, seems to resist quantum computers attacks. Moreover, the best existing algorithms solving them are exponential in time. The purpose of this thesis is the construction of public key cryptographic primitives for anonymity, whose security is based on the latter.

In particular, we are interested in ring signature schemes. First, we propose a new formal definition of anonymity and we present a new ring signature scheme. Second, we give a rigorous study of security, following two definitions of unforgeability. The first of which is unforgeability against chosen-subring attacks and the other one is unforgeability with respect to insider corruption.

Afterwards, we present a new ring identification scheme and we develop a full analysis of its security. Finally, we show that the techniques used to build this scheme, can be used to construct a threshold ring identification scheme.

Keywords. Public key cryptography, lattices, ring signature, ring identification, threshold ring identification, anonymity, provable security.

Table des matières

Table des figures

Introduction

En 1976, dans le célèbre article "*New Directions in Cryptography*" [DH76], Diffie et Hellman ont présenté une nouvelle façon de faire de la cryptographie. Plus précisément, ils ont introduit le concept de cryptographie asymétrique, aussi appelée à clé publique. Elle permet d'effectuer des communications sécurisées sans partage d'une même clé secrète. Pour cela, chaque utilisateur possède une paire de clés, une clé secrète (ou clé privée) connue seulement de l'utilisateur et une clé publique connue de tous. Le premier schéma de chiffrement asymétrique a été proposé par Rivest, Shamir et Adelman [RSA78], mieux connu sous le nom de RSA. En s'appuyant sur l'idée de la clé publique, Diffie et Hellman ont introduit le concept de signature numérique. Celui-ci permet d'avoir les mêmes propriétés que la signature manuscrite sur un papier. De manière informelle, un schéma de signature numérique prend en entrée un message à signer et retourne une chaîne de caractères qui relie le message à la clé publique du signataire. Étant donnés le message et la clé publique du signataire, il doit être possible à n'importe quelle personne tierce de vérifier la signature. D'autre part, seul l'utilisateur qui connaît la clé secrète (le signataire) doit être capable de générer une signature. Dans le même contexte de cryptographie à clé publique, Feige, Fiat, et Shamir ont montré, en 1988, qu'il est possible de construire des schémas d'identification permettant à une personne de donner une preuve tangible de son identité. D'une manière générale, le prouveur montre à un vérificateur qu'il connait la clé secrète associée à la clé publique sans donner aucune information sur le secret.

Durant les trentes dernières années, la cryptographie à clé publique n'a pas cessé de progresser en proposant des nouvelles primitives cryptographiques qui répondent aux exigences d'un monde numérique qui évolue rapidement. Par exemple, on peut imaginer un scénario dans lequel des utilisateurs collaborent ensemble pour signer numériquement un document. Le but est de signer le document au nom de tous les autres utilisateurs tout en restant anonyme. Un autre exemple consiste à permettre à un utilisateur de prouver son appartenance à une entité sans révéler son identité. C'est dans ce contexte que, dans ce mémoire, nous nous intéressons aux primitives cryptographiques à clé publique permettant la protection de la vie privée des usagers.

La cryptographie basée sur les réseaux euclidiens a été introduite par Ajtai en

1996. Elle est aujourd'hui l'un des domaines de recherche les plus actifs en cryptographie. Dans ces dix dernières années, ce domaine a inspiré beaucoup de chercheurs de la communauté cryptographique internationale aussi bien académique que industrielle. Des groupes industriels comme IBM et Microsoft ont financé des projets au sein de leurs laboratoires de recherche sur des thématiques liées à la cryptographie basée sur les réseaux, par exemple le chiffrement homomorphe [Gen09]. Ajtai [Ajt96] a montré que les instances aléatoires d'un certain problème sont aussi difficiles à résoudre que toutes les instances d'un problème considéré comme difficile standard des réseaux. Depuis lors, plusieurs primitives cryptographiques basées sur les réseaux ont été proposées : schémas de chiffrement [AD97, Reg09, SSTX09, PW11], schémas de signature [GPV08, Lyu09, Lyu12], schémas d'identification [MV03, Lyu08a, KTX08],... Baser la sécurité des primitives cryptographiques sur des problèmes difficiles des réseaux permet d'avoir une immunité contre les attaques utilisant les ordinateurs quantiques, contrairement aux problèmes de factorisation et de logarithme discret dont on sait qu'ils sont cassables par ce genre d'ordinateur [Sho97].

Réduction pire cas/cas moyen

Nous expliquons, de manière informelle, la réduction pire cas/cas moyen entre les problèmes difficiles issus des réseaux. Nous montrons l'utilité de cette réduction dans la construction des primitives cryptographiques par rapport aux outils utilisés dans la cryptographie basée sur des problèmes difficiles de la théorie des nombres.

En général, pour un problème donné, la complexité en cas moyen consiste à étudier la complexité des instances choisies aléatoirement. On dit qu'un problème est difficile dans le cas moyen, ou en moyenne, s'il n'existe pas d'algorithme polynomial, utilisant un aléa choisi de manière uniformément aléatoire, capable de résoudre avec une probabilité non négligeable une instance du problème choisie de manière uniformément aléatoire. D'autre part, la notion de complexité dans le pire cas consiste à étudier la complexité de toutes les instances du problème (même les plus difficiles). On dit qu'un problème est difficile dans le pire cas si pour toute instance fixée, il n'existe pas d'algorithme polynomial, utilisant un aléa choisi de manière uniformément aléatoire, capable de la résoudre avec une probabilité non négligeable.

Les schémas cryptographiques basés sur les réseaux ont une sécurité qui repose sur la difficulté des problèmes des réseaux dans le pire cas. Il est utile de rappeler que pour les schémas basés sur la théorie des nombres, la sécurité repose sur le cas moyen des problèmes difficiles. En prenant l'exemple du problème de factorisation des nombres, expliquons l'avantage de baser la sécurité des schémas cryptographiques sur des problèmes difficiles dans le pire cas : supposons que nous avons un schéma de signature numérique dont la sécurité est basée sur la

factorisation d'un entier N. Plus précisément, un adversaire qui réussi à attaquer le schéma de signature peut aussi trouver tous les facteurs premiers de N. Donc la sécurité de notre schéma dépend du problème de factorisation, pour cela on doit supposer que le problème est difficile à résoudre pour des N choisis aléatoirement. Malheureusement, ceci n'est pas le cas parce que l'on sait qu'il faut choisir N à partir d'une certaine distribution. En effet, N doit satisfaire certaines propriétés parce qu'il existe des algorithmes polynomiaux qui résolvent le problème de factorisation dans plusieurs cas. Même si on sait que N est difficile à factoriser lorsqu'il s'agit un produit de deux nombres premiers p et q qui sont suffisamment grands, il existe des conditions supplémentaires sur le choix de ces derniers. Par exemple, il faut que le plus grand facteur premier de $p-1$ et de $q-1$ soit très grand, $p+1$ et $q+1$ devraient avoir des facteurs premiers très grands, etc ... En revanche, la cryptographie basée sur les réseaux, n'a pas ce genre de problème, puisqu'on peut utiliser la réduction pire cas/cas moyen découverte par Ajtai [Ajt96]. Dans cette réduction, Ajtai a utilisé deux types de réseaux, que nous allons noter, réseaux de type A et réseaux de type B. Il a montré que trouver des vecteurs courts non nuls dans un réseau de type A choisi aléatoirement est aussi difficile que de trouver des vecteurs courts non nuls, même les plus difficiles à trouver, dans tous les réseaux de type B.

Par conséquent, il est plus facile de construire des primitives cryptographiques car il suffit de choisir des instances aléatoires du problème de trouver un vecteur court non nul sur des réseaux de type A et baser la sécurité du schéma dessus. Autrement dit, il faut montrer que casser le schéma implique la résolution de ce problème. La réduction pire cas/cas moyen nous permet d'avoir une assurance sur la sécurité du schéma puisqu'elle signifie qu'attaquer le schéma est aussi difficile que de résoudre des problèmes difficiles des réseaux de type B dans le pire cas.

Organisation du mémoire

Dans ce mémoire, nous proposons un schéma de signature ainsi que deux schémas d'identification pour l'anonymat. La sécurité de nos schémas repose sur des problèmes difficiles des réseaux. Notre première contribution est un schéma de signature de cercle qui permet de masquer l'identité du signataire au sein d'un groupe de signataires connus. Notre deuxième contribution consiste en un nouveau schéma d'identification de cercle qui permet à un utilisateur de prouver son appartenance à une entité sans révéler son identité. Nous généralisons ce résultat au cas où un ensemble d'utilisateurs veut donner une preuve d'appartenance à une entité d'une manière anonyme.

Dans le premier chapitre, nous fournissons, tout d'abord un rappel sur la distance statistique entre deux variables aléatoires et les notions de sécurité utilisées

en cryptographie. Nous donnons des définitions formelles des schémas cryptographiques qui seront utiles à la compréhension de ce mémoire. Ensuite, nous introduisons les notions de base des réseaux euclidiens et idéaux et nous rappelons quelques outils cryptographiques dont la sécurité repose sur des problèmes difficiles des réseaux euclidiens. Puis, nous décrirons les schémas d'identification de Lyubashevsky [Lyu08a, Lyu09] et de Kawachi, Tanaka, et Xagawa [KTX08], ainsi que les schémas de signature de Gentry, Peikert, et Vaikuntanathan [GPV08] et de Lyubashevsky [Lyu09, Lyu12].

Le deuxième chapitre se décompose en trois parties. Dans un premier temps nous donnons les définitions formelles d'un schéma de signature de cercle ainsi que de son modèle de sécurité. Dans un deuxième temps, nous rappelons deux schémas de signature de cercle dont la sécurité est basée sur des problèmes difficiles des réseaux. Finalement, nous présentons notre nouveau schéma de signature de cercle et nous analysons sa sécurité suivant deux modèles de sécurité différents. Ce résultat a fait l'objet d'une publication à la conférence Africacrypt [AMBB+13].

Dans le troisième chapitre, nous décrirons nos travaux sur les schémas d'identification anonymes. Ce chapitre est constitué de deux parties. La première partie sera consacrée à la présentation d'un nouveau schéma d'identification de cercle ainsi qu'à l'étude de sa sécurité. Nous comparons ses performances par le schéma proposé par Kawachi et al. [KTX08]. Dans la deuxième partie, nous montrons que la généralisation notre schéma d'identification de cercle permet d'obtenir un nouveau schéma d'identification de cercle à seuil. Les performances de notre nouveau schéma seront comparées à celles d'un schéma proposé par Cayrel, et al. [CLRS10b]. Ces résultats ont été présentés à la conférence PQCrypto'13 et publiés dans [BS13].

Chapitre 1

Les réseaux cryptographiques

Sommaire

Chapitre 1

Les réseaux cryptographiques

Le but de ce premier chapitre est d'introduire les notions mathématiques et les outils cryptographiques nécessaires pour la compréhension de nos schémas cryptographiques.

Dans la première section nous introduisons les notations que nous utiliserons dans ce mémoire. Dans la section 1.2, nous rappelons la distance statistique et ses propriétés. Dans la section 1.3, nous considérons les principes de sécurité utilisés en cryptographie. Dans la section 1.4, nous introduisons un premier ensemble d'outils cryptographiques qui seront utilisés dans ce mémoire. Dans la section 1.5, nous rappelons les fondements de la cryptographie basée sur les réseaux euclidiens et les réseaux idéaux ainsi qu'un deuxième ensemble d'outils cryptographiques. Dans la section 1.6, nous présentons trois schémas d'identification basés sur les réseaux. Dans la section 1.7, nous rappelons les schémas de signature numérique de Gentry, Peikert, et Vaikuntanathan et de Lyubashevsky.

1.1 Notations

Vecteurs, matrices, polynômes et vecteurs de polynômes. Nous représentons les vecteurs par des lettres romaines minuscules en gras italique $(\boldsymbol{a}, \boldsymbol{b}, \cdots)$ et les matrices par des lettres majuscules $(\boldsymbol{A}, \boldsymbol{B}, \cdots)$. Les polynômes sont notés par des lettres romaines et grecques en gras $(\mathbf{a}, \mathbf{b}, \cdots, \boldsymbol{\alpha}, \boldsymbol{\beta}, \cdots)$, en outre les vecteurs de polynômes sont donnés par des lettres avec des chapeaux $(\hat{\mathbf{a}}, \hat{\mathbf{b}}, \cdots, \hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\beta}}, \cdots)$. Soit m un entier positif tel que $\mathbf{a}_1, \cdots, \mathbf{a}_m$ soient des polynômes alors nous avons $\hat{\mathbf{a}} = (\mathbf{a}_1, \cdots, \mathbf{a}_m)$. La norme infinie d'un polynôme $\mathbf{a}$ est notée par $\|\mathbf{a}\|_\infty = \max_i |a^{(i)}|$, avec $a^{(i)}$ ses coefficients, la norme infinie d'un vecteur de polynômes $\hat{\mathbf{a}}$ est notée par $\|\hat{\mathbf{a}}\|_\infty = \max_i \|\mathbf{a}_i\|_\infty$. Nous utilisons l'opérateur $\|$ pour noter la concaténation.

Pour tout entier $\mathsf{p} \geq 1$, la norme ℓ_p d'un vecteur $\boldsymbol{x} = (x_1, \cdots, x_n)^T \in \mathbb{R}^n$ est notée par $\|\boldsymbol{x}\|_\mathsf{p}$ et correspond à $\left(\sum_{i=1}^n |x_i|^\mathsf{p}\right)^{1/\mathsf{p}}$. La norme ℓ_∞ est définie par

$\|\boldsymbol{x}\|_\infty = max_i|x_i|$. La norme ℓ_p d'une matrice $\boldsymbol{A} = [\boldsymbol{a}_1\|\cdots\|\boldsymbol{a}_m]$ est la norme du plus grand vecteur, $\|\boldsymbol{A}\|_\mathsf{p} = max_i\|\boldsymbol{a}_i\|_\mathsf{p}$. Nous notons le poids de Hamming de $\boldsymbol{x}$, c'est-à-dire le nombre des éléments non nuls de $\boldsymbol{x}$, par $w_H(\boldsymbol{x})$.

Ensembles. L'anneau quotient $\mathbb{Z}/p\mathbb{Z}$ sera noté par $\mathbb{Z}_p$, où p est un entier. Pour tout $n \in \mathbb{N}$, nous notons l'ensemble $\{1, \cdots, n\}$ par $[n]$ et par S_n l'ensemble de permutations de $[n]$. Soit S un ensemble fini, la notation $\alpha \leftarrow S$ signifie que α est choisi d'une manière uniformément aléatoire dans S. Le cardinal de S sera noté par $|S|$. Soient m, t deux entiers, nous notons l'ensemble des vecteurs dans $\{0,1\}^m$ ayant un poids de Hamming égal à t par $B(m,t)$.

Algorithmes. Pour tout algorithme A, la notation $\alpha \leftarrow$ A signifie que α est le résultat obtenu après la fin d'une exécution de l'algorithme A. Nous utiliserons les notations asymptotiques classiques $O, \Theta, \Omega, \omega, o$ [MVOV96]. La notation $\tilde{O}(n)$ correspond à $O(n\log^c n)$ pour toute constante c. Un algorithme est de temps polynomial si sa complexité en temps est de $O(n^c)$, où n est le paramètre de sécurité et c est une constante.

On dit qu'un problème P_1 se réduit à un problème P_2 si en utilisant une machine de Turing M_2 qui permet de résoudre le problème P_2, on peut construire une machine de Turing M_1 qui permet de résoudre le problème P_1 en utilisant M_2. Si la machine M_1 est en temps polynomial alors on dit que la réduction est polynomiale. Dans un tel cas, on dit que le problème P_2 est au moins aussi difficile que le problème P_1.

1.2 Probabilité et Distance statistique

Nous utilisons $\mathsf{poly}(\cdot)$ pour noter n'importe quel polynôme.

Définition 1.1 (Fonction négligeable)**.** *Une fonction* $\mathsf{negl} : \mathbb{N} \to \mathbb{R}^+$ *est dite négligeable si pour tout polynôme positif* $\mathsf{poly}(\cdot)$*, il existe* $n_0 \in \mathbb{N}$ *tel que pour tout* $n > n_0$*,* $\mathsf{negl}(n) \leq 1/\mathsf{poly}(n)$.

En se basant sur cette définition, nous définissons les notions de probabilités négligeables et écrasantes. Soient n un paramètre et $p(n)$ une probabilité dépendant de n. La probabilité $p(n)$ est dite négligeable si elle est une fonction négligeable de n. D'autre part, la probabilité $p(n)$ est dite écrasante si la probabilité $1 - p(n)$ est négligeable.

Définition 1.2 (Définition 8.5 dans [MG02])**.** *Soient* X *et* Y *deux variables aléatoires sur un ensemble* A*. La distance statistique* Δ *entre* X *et* Y *est définie par :*

$$\Delta(X,Y) = \frac{1}{2}\sum_{a\in A}|\mathsf{Pr}[X=a] - \mathsf{Pr}[Y=a]|.$$

Soit n un paramètre de sécurité, on dit que deux variables aléatoires X et Y dépendant de n sont statistiquement indistinguables si $\Delta(X, Y)$ est négligeable en n. On note ceci par $X \approx_S Y$.

Proposition 1.1 (Proposition 8.8 dans [MG02]). *Soient X, Y deux variables aléatoires sur un ensemble A et Z une troisième variable aléatoire sur un ensemble B (éventuellement différent de A). Si Z est indépendante de X et Y, alors*

$$\Delta((X, Z), (Y, Z)) = \Delta(X, Y).$$

Proposition 1.2 (Proposition 8.9 dans [MG02]). *Soient $X_1, \cdots, X_k$ et $Y_1, \cdots, Y_k$ deux listes de variables aléatoires indépendantes. Nous avons que*

$$\Delta((X_1, \cdots, X_k), (Y_1, \cdots, Y_k)) \leq \sum_{i=1}^{k} \Delta(X_i, Y_i).$$

Proposition 1.3 (Proposition 8.10 dans [MG02]). *Soient X et X' deux variables aléatoires sur un ensemble A. Pour toute fonction f (éventuellement randomisée) définies sur A, la distance statistique entre $f(X)$ et $f(X')$ est au plus*

$$\Delta(f(X), f(X')) \leq \Delta(X, X').$$

1.3 Sécurité prouvée

La sécurité prouvée a été introduite en 1984 par Goldwasser et Micali [GM84]. Elle permet d'analyser d'une manière formelle la sécurité d'une primitive cryptographique. Il y a deux piliers dans cette approche. Tout d'abord, il faut définir la sécurité, c'est-à-dire, ce qu'un schéma cryptographique doit satisfaire. Par exemple protéger l'intégrité d'un message dans le cas des schémas de signatures numériques ou bien l'impossibilité d'imiter un prouveur dans le cas des schémas d'identification. Ensuite, on examine si les buts définis préalablement sont atteints. Pour ceci, on calcule la probabilité qu'un attaquant gagne un jeu mené par un challengeur qui contrôle l'environnement de l'attaquant. Ce dernier peut avoir accès à des oracles simulés par le challengeur. Les tâches que l'attaquant doit accomplir pour gagner un jeu sont différentes d'une primitive à l'autre. Par exemple elles peuvent être liées à la capacité d'un attaquant à contrefaire des signatures ou bien à distinguer entre deux distributions de probabilités. Un tel jeu entre un challengeur et un attaquant est appelé un modèle de sécurité.

En pratique, on montre qu'un schéma cryptographique est sûr suivant un modèle de sécurité donné en utilisant une réduction à un problème jugé difficile. Une réduction de sécurité est similaire au principe de la réduction algorithmique des problèmes. Elle consiste à montrer que si un attaquant est capable de gagner

un jeu, alors il pourrait être utilisé par un autre algorithme afin de résoudre un problème difficile.

Modèle de l'oracle aléatoire. Le modèle de l'oracle aléatoire ou ROM (pour *Random Oracle Model*, en anglais) a été introduit par Bellare et Rogaway en 1993 [BR93]. Ce modèle suppose que les fonctions de hachage (voir Définition 1.15) se comportent comme des fonctions aléatoires. Cela implique que leurs sorties sont indistinguables d'une suite de valeurs aléatoires. Prouver la sécurité d'une primitive cryptographique dans ce modèle consiste à remplacer la fonction de hachage du schéma en question par un oracle aléatoire. Pour chaque requête envoyée, l'oracle répond avec un haché qui suit une distribution uniforme dans l'espace de hachage et il enregistre le couple (requête/réponse) dans une liste. Puisque l'oracle simule une fonction de hachage, lorsqu'il est appelé avec la même requête plusieurs fois, il répondra toujours avec la même valeur.

Modèle standard. Le modèle standard ne pose aucune hypothèse sur les fonctions de hachage utilisées dans une primitive cryptographique. Les preuves de sécurité dans ce modèle donnent des propriétés de sécurité plus fortes que celles obtenues dans le modèle de l'oracle aléatoire.

1.4 Primitives cryptographiques

1.4.1 Fonctions à sens unique

Une fonction à sens unique est une fonction qui est facile à calculer mais difficile à inverser.

Définition 1.3 ([Gol07])**.** *Une fonction $f : \{0,1\}^\star \to \{0,1\}^\star$ est dite à sens unique si elle satisfait les deux conditions suivantes :*

- *Facile à calculer : il existe un algorithme polynomial* A *qui pour tout $x \in \{0,1\}^\star$ retourne $f(x)$.*
- *Difficile à inverser : pour tout algorithme probabiliste polynomial* A' *et pour tout n suffisamment grand, si $x \leftarrow U_n$ alors*

$$\mathsf{Pr}\left[\mathsf{A}'(f(x), 1^n) \in f^{-1}(f(x))\right] < \frac{1}{\mathsf{poly}(n)},$$

où U_n désigne la distribution uniforme sur $\{0,1\}^n$.

1.4.2 Schémas d'engagement

Le but d'un schéma d'engagement est de permettre à un émetteur de s'engager sur une valeur auprès d'un destinataire. Au début l'émetteur calcule un engagement sans révéler la valeur engagée. Après cette opération le destinataire aura

une garantie que l'émetteur ne pourra plus la modifier. Pour vérifier l'engagement, l'émetteur révèle toutes les informations nécessaires qui permettent au destinataire de s'assurer qu'il s'agit bien de la valeur engagée au départ.

Nous donnons une définition formelle d'un schéma d'engagement ainsi que de ses propriétés de sécurité.

Définition 1.4. *[BCC88] Un schéma d'engagement est défini par les trois algorithmes suivants :*

$\mathsf{param} \leftarrow \mathsf{Init}(1^n)$: Init *est un algorithme probabiliste polynomial qui prend en entrée un paramètre de sécurité* 1^n *et retourne des paramètres publics* param.

$(c, r) \leftarrow \mathsf{Commit}(\mathsf{param}, \mu)$: Commit *est un algorithme probabiliste polynomial d'engagement qui prend en entrée les paramètres publics* param *et un message* $\mu \in \{0,1\}^\star$. *Il renvoie un engagement* c *et une valeur aléatoire* r *nécessaire pour la vérification.*

$\{0,1\} \leftarrow \mathsf{Verif}(\mathsf{param}, \mu, c, r)$: Verif *est un algorithme probabiliste polynomial qui prend en entrée les paramètres publics* param*, le message* μ*, l'engagement* c *et la valeur aléatoire* r. *Il renvoie* 1 *si* c *est bien un engagement sur* μ*, sinon il renvoie* 0.

Modèle de sécurité. Un schéma d'engagement doit satisfaire les deux notions de sécurité suivantes :

— Indistinguabilité (Hiding) : le destinataire ne doit rien apprendre sur le message μ s'il ne dispose que de l'engagement c. En effet, pour tout couple de message $\mu_0, \mu_1 \in \{0,1\}^\star$, il doit être infaisable de distinguer entre l'ensemble des engagements sur μ_0 et l'ensemble des engagements sur μ_1.

— Engagement (Binding) : l'émetteur ne doit pas être capable de changer le message engagé après avoir généré un engagement sur ce dernier. En effet, pour tout message $\mu_0 \in \{0,1\}^\star$ et pour tout $(c, r_0) \leftarrow \mathsf{Commit}(\mathsf{param}, \mu_0)$, il doit être infaisable de trouver $\mu_1 \neq \mu_0$ et r_1, tels que

$$\mathsf{Verif}(\mathsf{param}, \mu_0, c, r_0) = \mathsf{Verif}(\mathsf{param}, \mu_1, c, r_1) = 1.$$

1.4.3 Preuves de connaissances

Le concept des preuves de connaissances (*proof of knowledge*) a été introduit par Goldwasser, Micali et Rackoff [GMR85] et formalisé en 1988 par Feige, Fiat et Shamir [FFS88]. Une preuve de connaissance est un protocole interactif dans lequel un prouveur $\mathcal{P}$ essaye de convaincre un vérificateur $\mathcal{V}$ qu'il connait un secret. Pendant l'exécution du protocole, le prouveur ne divulgue pas son secret au vérificateur et à la fin ce dernier obtient une garantie que le prouveur connaît bien le secret.

La notion de preuve sans divulgation (*zero-knowledge*) a été introduite par Goldwasser, Micali et Rackoff [GMR85]. Elle permet d'évaluer la sécurité du prouveur lorsqu'il exécute un protocole interactif avec un vérificateur. Cette notion permet de démontrer qu'aucune information sur le secret du prouveur ne pourrait être obtenue par le vérificateur même si ce dernier est malhonnête.

Définition 1.5. *[Bre02][Algorithme interactif] Un algorithme interactif $\mathcal{P}$ est un algorithme probabiliste polynomial qui prend en entrée un message μ_{in} et un état d'information st. Il retourne un message μ_{out} et une mise à jour de st. Nous utilisons la notation $(\mu_{out}, st') \leftarrow \mathcal{P}(\mu_{in}, st)$.*

Définition 1.6. *[Bre02][Protocole interactif] Un protocole interactif consiste en deux d'algorithmes interactifs $\mathcal{P}$ et $\mathcal{V}$ tels que les messages de type μ_{out} de $\mathcal{P}$ sont des messages de type μ_{in} pour $\mathcal{V}$ et réciproquement. Le protocole interactif entre $\mathcal{P}$ et $\mathcal{V}$ se termine lorsque $\mathcal{V}$ donne en sortie l'état* Accepte *ou* Rejette*. Nous notons l'interaction entre $\mathcal{P}$ et $\mathcal{V}$ par :*

$$(\mathsf{T}, \mathsf{desc}) \leftarrow \mathsf{Run}[(\mathcal{P}(p) \leftrightarrow \mathcal{V}(v))(w)]$$

avec p (respectivement ,v) une entrée auxiliaire pour $\mathcal{P}$ (respectivement, $\mathcal{V}$) et w une entrée en commun pour les deux. Les échanges entre $\mathcal{P}$ et $\mathcal{V}$ sont notés par T *et la décision de $\mathcal{V}$ est définie par la variable* desc $\in \{0,1\}$*, qui vaut 1 si ce dernier termine dans un état* Accepte *et 0 s'il termine dans un état* Rejette*.*

1.4.3.1 Preuve de connaissance

Soit $Q(\cdot,\cdot) \rightarrow \{0,1\}$, un prédicat à deux variables. Nous appelons y un témoin de x par le prédicat Q si $Q(y,x) = 1$. Un protocole de preuve de connaissance est un protocole interactif entre un prouveur $\mathcal{P}$ et un vérificateur $\mathcal{V}$ qui partagent une entrée publique x. Le but de cette interaction est de permettre au prouveur de montrer qu'il connaît un témoin y tel que $Q(y,x) = 1$.

Un protocole interactif de preuve de connaissance doit satisfaire deux propriétés. Premièrement, il doit satisfaire la propriété de consistance qui assure qu'un prouveur honnête doit être toujours accepté par le vérificateur avec une probabilité écrasante. Deuxièmement, il doit satisfaire la propriété de significativité qui assure que si un prouveur malhonnête $\tilde{\mathcal{P}}$ est accepté par le vérificateur avec une probabilité non négligeable, alors il existe un extracteur qui peut trouver le témoin de $\mathcal{P}$.

Nous allons maintenant donner des définitions formelles de ce qu'est une preuve de connaissance, une preuve de connaissance sans divulgation et une preuve à témoin indistinguable.

Définition 1.7. *[Bre02][Preuve de connaissance] Un protocole interactif $(\mathcal{P} \leftrightarrow \mathcal{V})$ est appelé une preuve de connaissance d'un témoin y de x, s'il satisfait les propriétés suivantes :*

Consistance *: Si $\mathcal{P}$ est un prouveur honnête alors il est accepté par le vérificateur avec une probabilité proche de 1. Autrement dit, si $Q(y,x)=1$ alors,*

$$\mathsf{Pr}[\mathsf{desc}=1:(\mathsf{T},\mathsf{desc})\leftarrow\mathsf{Run}[(\mathcal{P}(y)\leftrightarrow\mathcal{V})(x)]] \text{ est écrasante.}$$

Significativité *: Soit $\tilde{\mathcal{P}}$ un prouveur malhonnête. Il existe un extracteur de connaissance $\mathcal{E}$ (algorithme probabiliste polynomial) tel que si*

$$\mathsf{Pr}[\mathsf{desc}=1:(\mathsf{T},\mathsf{desc})\leftarrow\mathsf{Run}[(\tilde{\mathcal{P}}\leftrightarrow\mathcal{V})(x)]] \text{ est non négligeable,}$$

alors

$$\mathsf{Pr}[Q(\mathcal{E}^{\tilde{\mathcal{P}}}(x),x)=1] \text{ est non négligeable.}$$

$\mathcal{E}^{\tilde{\mathcal{P}}}(x)$ désignant la valeur retournée par $\mathcal{E}$ en lui donnant accès à $\tilde{\mathcal{P}}$ et x.

Maintenant nous rappelons la notion de vue du vérificateur qui sera nécessaire aux définitions de preuve sans divulgation et de preuves à témoin indistinguable.

Définition 1.8. *[Bre02][La vue du vérificateur] Soit $(\mathcal{P}\leftrightarrow\mathcal{V})$ un protocole interactif tel que w est une entrée en commun pour les deux algorithmes interactifs. Les entrées auxiliaires pour $\mathcal{P}$ et $\mathcal{V}$ sont respectivement p et v. La vue du vérificateur est la distribution des variables aléatoires décrivant les messages échangés entre $\mathcal{P}$ et $\mathcal{V}$ pendant le déroulement du protocole. Nous utilisons la notation :*

$$\mathsf{Vue}_{\mathcal{V}}\left[(\mathcal{P}(p)\leftrightarrow\mathcal{V}(v))(w)\right].$$

1.4.3.2 Preuves sans divulgation

Nous considérons un protocole interactif de connaissance $(\mathcal{P}\leftrightarrow\mathcal{V})$ durant lequel $\mathcal{P}$ essaie de convaincre $\mathcal{V}$ qu'il connaît un témoin y lié à x par un prédicat Q (c'est-à-dire tel que $Q(y,x)=1$).

Définition 1.9. *[Bre02][Preuve sans divulgation] Le protocole $(\mathcal{P}\leftrightarrow\mathcal{V})$ est dit statistiquement sans divulgation, si pour tout vérificateur (potentiellement malhonnête) $\tilde{\mathcal{V}}$, il existe un algorithme probabiliste polynomial $\mathcal{S}_{\tilde{\mathcal{V}}}$ (un simulateur) tel que :*

$$\overline{\mathcal{S}_{\tilde{\mathcal{V}}}(x)}\approx_S \mathsf{Vue}_{\tilde{\mathcal{V}}}\left[(\mathcal{P}(y)\leftrightarrow\tilde{\mathcal{V}})(x)\right].$$

Où $\overline{\mathcal{S}_{\tilde{\mathcal{V}}}}$ désigne la distribution des variables aléatoires générés par le simulateur $\mathcal{S}_{\tilde{\mathcal{V}}}$.

L'intuition derrière la définition est qu'un vérificateur (potentiellement malhonnête) $\tilde{\mathcal{V}}$ n'acquiert aucune information sur le témoin de $\mathcal{P}$. En effet, le simulateur $\mathcal{S}$ qui a accès aux mêmes données que $\tilde{\mathcal{V}}$ (c'est-à-dire, il ne connaît pas le secret y de $\mathcal{P}$) est capable de construire des variables dont la distribution est statistiquement proche de celle d'une interaction entre $\tilde{\mathcal{V}}$ et $\mathcal{P}$.

Remarque 1.1. *Nous définissons une notion de preuve sans divulgation plus faible que celle donnée dans la Définition 1.9, c'est la notion de preuve sans divulgation face à un vérificateur honnête. Dans cette configuration, il existe un simulateur qui peut construire une transcription statistiquement proche d'une instance du protocole jouée par des parties honnêtes ($\mathcal{V}$ n'est plus malhonnête comme nous l'avons supposé dans la Définition 1.9). Plus formellement, un protocole interactif $(\mathcal{P} \leftrightarrow \mathcal{V})$ est dit statistiquement sans divulgation face à un vérificateur honnête s'il existe un simulateur $\mathcal{S}_{\mathcal{V}}$ tel que :*

$$\overline{\mathcal{S}_{\mathcal{V}}(x)} \approx_S \mathsf{Vue}_{\mathcal{V}}\left[(\mathcal{P}(y) \leftrightarrow \mathcal{V})(x)\right].$$

Où $\overline{\mathcal{S}_{\mathcal{V}}(x)}$ désigne la distribution des variables aléatoires générés par le simulateur $\mathcal{S}_{\mathcal{V}}$.

Remarque 1.2. *En 1998, Goldreich, Sahai et Vadhan ont démontré dans [GSV98], que les preuves de connaissance sans divulgation face à un vérificateur honnête peuvent être transformées en des preuves de connaissance sans divulgation suivant la Définition 1.9.*

1.4.3.3 Preuves à témoin indistinguable

Une variante de la notion de sans divulgation est la notion de témoin indistinguable (*witness indistinguishability*) introduite par Feige et Shamir [FS90] en 1990. Pour un protocole interactif à témoin indistinguable il existe toujours plusieurs témoins pour le prouveur. Intuitivement, lorsqu'on considère la paire de témoins (y_1, y_2) de x (c'est-à-dire que $Q(y_1, x) = Q(y_2, x) = 1$), un protocole interactif est à témoin indistinguable s'il est impossible pour tout vérificateur (potentiellement malhonnête) de distinguer entre deux exécutions du protocole où le prouveur utilise un témoin de x parmi (y_1, y_2) à chaque fois, même si le vérificateur connaît la paire (y_1, y_2). D'une manière formelle nous avons la définition suivante de cette propriété.

Définition 1.10 (Témoin indistinguable)**.** *Un protocole interactif $(\mathcal{P} \leftrightarrow \mathcal{V})$ est statistiquement à témoin indistinguable si pour tout vérificateur malhonnête $\tilde{\mathcal{V}}$ et pour toute paire de témoins (y_1, y_2) de x, nous avons que :*

$$\mathsf{Vue}_{\tilde{\mathcal{V}}}\left[(\mathcal{P}(y_1) \leftrightarrow \tilde{\mathcal{V}}(z))(x)\right] \approx_S \mathsf{Vue}_{\tilde{\mathcal{V}}}\left[(\mathcal{P}(y_2) \leftrightarrow \tilde{\mathcal{V}}(z))(x)\right].$$

Avec z, une entrée auxiliaire pour $\tilde{\mathcal{V}}$ qui peut prendre la valeur $y_1 \| y_2$.

Proposition 1.4. *Nous avons les résultats suivants de Feige et Shamir [FS90] :*

1. *Soit $(\mathcal{P}, \mathcal{V})$ un protocole interactif. Si le protocole est sans divulgation, alors il est aussi à témoin indistinguable. La réciproque est fausse.*
2. *La propriété de témoin indistinguable est préservée en cas d'exécution de plusieurs instances du même protocole en parallèle.*

1.4.4 Schémas d'identification

Définition 1.11 (Schéma d'identification [AABN02])**.** *Un schéma d'identification consiste en un triplet d'algorithmes* $(\mathsf{Id{-}init}, \mathsf{Id{-}keygen}, (\mathcal{P} \leftrightarrow \mathcal{V}))$ *:*

- $\mathsf{Id{-}init}(1^n)$: *est un algorithme probabiliste polynomial d'initialisation, il prend en entrée un paramètre de sécurité* 1^n *et retourne un paramètre public* param*.*
- $\mathsf{Id{-}keygen}(\mathsf{param})$: *est un algorithme probabiliste polynomial qui prend en entrée un paramètre public* param *et retourne une paire de clés* (pk, sk)*.*
- $(\mathcal{P}(sk) \leftrightarrow \mathcal{V})(\mathsf{param}, pk)$: *est un protocole interactif entre le prouveur* $\mathcal{P}$ *qui prend en entrée* (pk, sk, param) *et le vérificateur qui prend en entrée* (pk, param)*. A la fin de l'interaction, le vérificateur* $\mathcal{V}$ *retourne* $\mathsf{desc} \in \{0, 1\}$ *qui vaut* 1 *si ce dernier accepte et* 0 *s'il rejette.*

De plus, un prouveur honnête $\mathcal{P}$ *doit toujours être en mesure de réussir à convaincre le vérificateur. D'une manière formelle, pour tous* n*,* $\mathsf{param} \in \mathsf{Id{-}init}(1^n)$*,* $(pk, sk) \in \mathsf{Id{-}keygen}(\mathsf{param})$*, on a*

$$\Pr\left[\mathsf{desc} = 1 : \mathsf{desc} \leftarrow (\mathcal{P}(sk) \leftrightarrow \mathcal{V})(\mathsf{param}, pk)\right] = 1.$$

Modèle de sécurité des protocoles d'identification. Nous considérons deux modèles de sécurité : la sécurité contre les attaques concurrentes ou bien parallèles et la sécurité contre les attaques actives. La sécurité contre les attaques concurrentes est définie par deux phases : une phase d'apprentissage et une phase d'imposture. Pendant la première phase, l'attaquant joue le rôle d'un vérificateur malhonnête et il interagit avec un challengeur qui joue le rôle d'un prouveur honnête. Pendant cette phase l'attaquant peut lancer plusieurs instances du protocole d'une manière parallèle. Dans chaque instance, le challengeur utilise le même secret mais il génère d'une manière indépendante les valeurs aléatoires nécessaires pour l'exécution du protocole. Dans la deuxième phase les deux protagonistes (challengeur et l'attaquant) inversent leur rôle. L'attaquant va essayer de démontrer au challengeur qu'il peut se faire passer pour un prouveur.

Dans l'approche de sécurité contre les attaques actives, l'attaquant ne pourrait interagir que d'une manière séquentielle avec le prouveur durant la phase d'apprentissage. Plus formellement nous avons les deux définitions suivantes :

Définition 1.12. *[BP02] Soit un schéma d'identification défini par les algorithmes* $(\mathsf{Id{-}init}, \mathsf{Id{-}keygen}, (\mathcal{P} \leftrightarrow \mathcal{V}))$*. On considère le jeu suivant entre un challengeur* $\mathcal{C}$ *et un attaquant* $\mathcal{I}$*.*

- *Le challengeur est modélisé par un couple d'algorithmes interactifs* $(\mathcal{P}, \mathcal{V})$*.*
- *L'attaquant est modélisé par un couple d'algorithmes interactifs* $(\tilde{\mathcal{P}}, \tilde{\mathcal{V}})$*.*

Phase de préparation : *Soit* n *le paramètre de sécurité. Le challengeur obtient* $\mathsf{param} \leftarrow \mathsf{Id{-}init}(1^n)$ *et* $(pk, sk) \leftarrow \mathsf{Id{-}keygen}(\mathsf{param})$*. Ensuite, il envoie* $(1^n, \mathsf{param}, pk)$ *à l'attaquant.*

Phase d'apprentissage : *Pendant cette phase l'attaquant $\tilde{\mathcal{V}}$ peut envoyer des requêtes à l'oracle suivant :*

— Prouv. *Il prend en entrée un message μ_{in}. Il initialise l'état du prouveur par $st_{\mathcal{P}} \leftarrow (\mathsf{param}, sk)$. Puis il retourne $\mu_{out} \leftarrow \mathcal{P}(\mu_{in}, st_{\mathcal{P}})$. Bien sûr, l'oracle* Prouv *lance des instances en parallèle du prouveur $\mathcal{P}$ quand ceci est demandé par l'attaquant $\mathcal{V}$.*

Phase de challenge : *À un certain moment, le challengeur obtient*

$$(\mathsf{T}, \mathsf{desc}) \leftarrow \mathsf{Run}[(\tilde{\mathcal{P}} \leftrightarrow \mathcal{V})(\mathsf{param}, pk)].$$

L'attaquant $\mathcal{I} = (\tilde{\mathcal{P}}, \tilde{\mathcal{V}})$ gagne le jeu si $\mathsf{desc} = 1$. *Un schéma d'identification est sûr contre l'usurpation d'identité avec attaques parallèles, si la probabilité qu'un attaquant $\mathcal{I}$ puisse gagner ce jeu est négligeable en le paramètre de sécurité n.*

Définition 1.13. *[FFS88] Soit un schéma d'identification défini par les algorithmes $(\mathsf{Id{-}init}, \mathsf{Id{-}keygen}, (\mathcal{P} \leftrightarrow \mathcal{V}))$. On considère le jeu suivant entre un challengeur $\mathcal{C}$ et un attaquant $\mathcal{I}$.*

— *Le challengeur est modélisé par un couple d'algorithmes interactifs $(\mathcal{P}, \mathcal{V})$.*
— *L'attaquant est modélisé par un couple d'algorithmes interactifs $(\tilde{\mathcal{P}}, \tilde{\mathcal{V}})$.*

Phase de préparation : *Soit n le paramètre de sécurité. Le challengeur obtient* $\mathsf{param} \leftarrow \mathsf{Id{-}init}(1^n)$ *et* $(pk, sk) \leftarrow \mathsf{Id{-}keygen}(\mathsf{param})$. *Ensuite, il envoie $(1^n, \mathsf{param}, pk)$ à l'attaquant.*

Phase d'apprentissage : *Pendant cette phase l'attaquant $\tilde{\mathcal{V}}$ peut envoyer des requêtes à l'oracle suivant :*

— Prouv. *Il prend en entrée un message μ_{in}. Il initialise l'état du prouveur par $st_{\mathcal{P}} \leftarrow (\mathsf{param}, sk)$. Puis il retourne $\mu_{out} \leftarrow \mathcal{P}(\mu_{in}, st_{\mathcal{P}})$.*

Phase de challenge : *À un certain moment, le challengeur obtient*

$$(\mathsf{T}, \mathsf{desc}) \leftarrow \mathsf{Run}[(\tilde{\mathcal{P}} \leftrightarrow \mathcal{V})(\mathsf{param}, pk)].$$

L'attaquant $\mathcal{I} = (\tilde{\mathcal{P}}, \tilde{\mathcal{V}})$ gagne le jeu si $\mathsf{desc} = 1$. *Un schéma d'identification est sûr contre l'usurpation d'identité avec attaques actives, si la probabilité qu'un attaquant $\mathcal{I}$ puisse gagner ce jeu est négligeable en le paramètre de sécurité n.*

Dans [FFS88], Feige, Fiat, et Shamir ont montré comment on peut obtenir un protocole d'identification à partir d'une preuve de connaissance sans divulgation. De manière générale, le prouveur génère une paire de clés (pk, sk). Avant de commencer la session d'identification, le prouveur donne sa clé publique au vérificateur. Ce dernier va vérifier que le prouveur est en possession de la clé sk qui correspond à la clé pk. Le prouveur convainc le vérificateur qu'il connaît la clé sk sans la révéler. En particulier, durant le protocole le prouveur ne donne aucune information sur son secret.

Exemple : En se basant sur les techniques données dans [FFS88], Goldreich [Gol07] a donné quelques exemples des schémas d'identification basés sur les preuves de connaissance. Ici nous rappelons une construction générique (Construction 4.7.9 dans [Gol07]) de protocole d'identification basé sur une fonction à sens unique f (voir Définition 1.3) :

— La paire de clés est générée de la manière suivante : le prouveur choisit $\boldsymbol{x} \in \{0,1\}^k$ (k étant un paramètre de sécurité) et il calcule $\boldsymbol{y} = f(\boldsymbol{x})$. La clé publique est $\boldsymbol{y}$ et la clé secrète est $\boldsymbol{x}$ (le prouveur partage sa clé publique avec le vérificateur avant le processus d'identification).

— Pour s'identifier auprès du vérificateur, le prouveur donne une preuve de connaissance d'un $\boldsymbol{x}$ tel que $f(\boldsymbol{x}) = \boldsymbol{y}$.

Proposition 1.5 (Proposition 4.7.10 dans [Gol07])**.** *Si f est une fonction à sens unique et si la preuve de connaissance utilisée est sans divulgation, alors la construction ci-dessus constitue un schéma d'identification.*

Par conséquent, les schémas d'identification existent si les fonctions à sens unique existent.

Définition 1.14 (Schéma d'identification en trois passes)**.** *Un schéma d'identification en trois passes consiste en un triplet d'algorithmes* $(\mathsf{Id-init}, \mathsf{Id-keygen}, (\mathcal{P} \leftrightarrow \mathcal{V}))$ *:*

$\mathsf{Id-init}(1^n)$: *est un algorithme probabiliste polynomial d'initialisation, il prend en entrée un paramètre de sécurité* 1^n *et retourne un paramètre public* param.

$\mathsf{Id-keygen}(\mathsf{param})$: *est un algorithme probabiliste polynomial qui prend en entrée un paramètre public* param *et retourne une paire de clés* (pk, sk).

$(\mathcal{P} \leftrightarrow \mathcal{V})$: *est un protocole interactif entre le prouveur $\mathcal{P}$ qui prend en entrée* (pk, sk, param) *et le vérificateur qui prend en entrée* (pk, param)*. Le protocole se passe en trois étapes :*

— *[Étape d'engagement] : $\mathcal{P}$ calcule un engagement et l'envoie à $\mathcal{V}$.*

— *[Étape de challenge] : $\mathcal{V}$ répond par un challenge.*

— *[Étape de réponse] : $\mathcal{P}$ envoie une réponse au challenge de $\mathcal{V}$, qui permettra à ce dernier de s'assurer que $\mathcal{P}$ connait bien le secret sk.*

Dans la littérature [Poi95, SSH11, CVA11], on peut trouver des schémas d'identification en cinq passes. Ces schémas sont une variation des schémas en trois passes. Ils suivent le même modèle donné dans la Définition 1.14 mais avec deux phases de challenges et deux phases de réponses.

1.4.5 Fonctions de hachage et le *Leftover Hash Lemma*

En cryptologie, une fonction de hachage h est une fonction à sens unique particulière. Elle est définie d'un ensemble D_n dans un ensemble R_n, où n est un

paramètre de sécurité, et elle permet de compresser les données. Pour un message quelconque dans D_n, elle permet d'obtenir une chaine de bits de taille fixe appartenant à R_n. La sécurité d'une fonction de hachage est résumée dans trois propriétés que nous décrirons d'une manière informelle.

Définition 1.15. *On dit que h est une fonction de hachage cryptographiquement sûre 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 un message $\mu \in D_n$, il doit être calculatoirement difficile de trouver un message $\mu' \neq \mu$ tel que $h(\mu) = h(\mu')$.*
- *résistance aux collisions : Il est calculatoirement difficile de trouver deux messages μ, μ' dans D_n tels que $h(\mu) = h(\mu')$.*

Dans ce mémoire nous utilisons des familles de fonctions de hachage. Soit n le paramètre de sécurité, une famille de fonction de hachage $\mathcal{H}$ consiste en un ensemble de fonctions de hachage h_{param} définies de l'ensemble D_n dans R_n et indexées par un paramètre $\mathsf{param} \in P_n$ qui permet de définir chaque fonction, où P_n est un ensemble qui dépend de n. Nous utilisons la notation $\mathcal{H} = \{h_{\mathsf{param}} : D_n \to R_n\}$.

Dans la suite, nous manipulerons des familles de fonctions de hachage ayant des fonctions définies à partir d'une matrice ou bien à partir d'un vecteur de polynômes.

Maintenant, nous rappelons le *leftover hash lemma* [HILL99] qui quantifie la distance statistique entre la distribution uniforme sur $\mathcal{H} \times R_n$ et un couple $(h_{\mathsf{param}}, h_{\mathsf{param}}(\mu))$ où h_{param}, μ sont choisis d'une manière uniforme. Plus formellement, nous avons le lemme suivant.

Lemme 1.1 (*Leftover Hash Lemma*)**.** *Soit une famille de fonctions de hachage $\mathcal{H} = \{h_{\mathsf{param}} : D_n \to R_n | \mathsf{param} \in P_n\}$. Soient R la distribution uniforme sur R_n, P la distribution uniforme sur P_n,* param *et μ deux variables aléatoires qui suivent la distribution uniforme sur $P \times D_n$, nous avons*

$$\Delta((\mathsf{param}, h_{\mathsf{param}}(\mu)), (P, R)) \leq \frac{1}{2}\sqrt{\frac{|R_n|}{|D_n|}}.$$

1.4.6 Signatures numériques

La notion de signature numérique a été introduite par Diffie et Hellman dans [DH76] puis formalisée, en 1988, par Goldwasser, Micali et Rivest [GMR88]. Les buts d'une signature numérique sont d'assurer :

- L'authenticité du signataire.
- L'intégrité des messages (qui permet de s'assurer qu'un message n'a pas été modifié lors de sa transmission).

— La non-répudiation (qui permet de s'assurer que le signataire ne puisse pas nier sa propre signature).

Définition 1.16 (Schéma de signature [GMR88]). *Un schéma de signature numérique consiste en un quadruplet d'algorithmes (*$\mathsf{S-init}$, $\mathsf{S-keygen}$, $\mathsf{S-sign}$,$\mathsf{S-verify}$*) :*

$\mathsf{param} \leftarrow \mathsf{S-init}(1^n)$: *est un algorithme probabiliste polynomial d'initialisation, il prend en entrée un paramètre de sécurité* 1^n *et retourne un paramètre public qu'on notera* param.

$(pk, sk) \leftarrow \mathsf{S-keygen}(\mathsf{param})$: *est un algorithme probabiliste polynomial qui prend en entrée le paramètre* param *et retourne une paire de clés* (pk, sk).

$\sigma \leftarrow \mathsf{S-sign}(\mathsf{param}, (pk, sk), \mu)$: *est un algorithme probabiliste polynomial qui prend en entrée le paramètre public* param*, une paire de clés* (pk, sk) *et un message* $\mu \in \{0,1\}^\star$ *à signer et retourne une signature* σ *de message suivant la clé secrète* sk.

$\{0,1\} \leftarrow \mathsf{S-verify}(\mathsf{param}, pk, \sigma, \mu)$: *est un algorithme polynomial déterministe qui prend en entrée le paramètre public* param*, la clé publique* pk*, la signature* σ *du message* μ*. Il retourne* 1 *si* σ *est une signature valide de* μ *et* 0 *sinon.*

Nous avons aussi la propriété de complétude suivante : Pour tous n, $\mathsf{param} \in \mathsf{S-init}(1^n)$, $(pk, sk) \leftarrow \mathsf{S-keygen}(\mathsf{param})$, $\mu \in \{0,1\}^\star$, $\sigma \in \mathsf{S-sign}(\mathsf{param}, (pk, sk), \mu)$ *on a,* $\mathsf{S-verify}(\mathsf{param}, pk, \sigma, \mu) = 1$.

Un schéma de signature numérique doit satisfaire la propriété de résistance à la contrefaçon existentielle contre les attaques à messages choisis [GMR88]. D'une manière informelle, cela signifie qu'un attaquant $\mathcal{F}$ (falsificateur) auquel est donné des signatures de messages de son choix ne devrait pas être en mesure de produire une signature d'un nouveau message. Plus formellement, nous avons le jeu suivant entre un challengeur $\mathcal{C}$ et un falsificateur $\mathcal{F}$.

Définition 1.17 ([GMR88]). *Soient un schéma de signature numérique défini par les algorithmes (*$\mathsf{S-init}$, $\mathsf{S-keygen}$, $\mathsf{S-sign}$,$\mathsf{S-verify}$*) et* n *un paramètre de sécurité. On considère le jeu suivant entre un challengeur* $\mathcal{C}$ *et un falsificateur* $\mathcal{F}$.

Phase d'initialisation : *Le challengeur exécute l'algorithme* $\mathsf{S-init}(1^n)$ *pour obtenir les paramètres* param*. Ensuite il génère une paire de clés grâce à l'algorithme* $\mathsf{S-keygen}$ *(*$(pk, sk) \leftarrow \mathsf{S-keygen}(\mathsf{param})$*). Le challengeur* $\mathcal{C}$ *initialise l'attaquant, en lui donnant* $(1^n, \mathsf{param}, pk)$.

Phase de requêtes : *Pendant cette phase l'attaquant peut envoyer des requêtes à l'oracle de signature. L'oracle reçoit* μ *(un message) comme requête et retourne la signature* $\sigma \leftarrow \mathsf{S-sign}(\mathsf{param}, (pk, sk), \mu)$.

Phase de réponse : $\mathcal{F}$ *retourne un message* $\mu^\star$ *et une signature* $\sigma^\star$*. L'attaquant gagne le jeu si* $\mathsf{S-verify}(\mathsf{param}, pk, \sigma^\star, \mu^\star) = 1$*, et le paire* $(\mu^\star, \sigma^\star)$ *ne fait pas partie des paires générées pendant la phase de requêtes.*

Un schéma de signature satisfait la propriété de résistance à la contrefaçon existentielle contre les attaques à messages choisis si la probabilité qu'un attaquant $\mathcal{F}$ gagne ce jeu est négligeable en le paramètre de sécurité n.

1.5 Réseaux et applications

1.5.1 Les réseaux euclidiens

Un réseau est un sous groupe discret de $\mathbb{R}^d$, où d est un entier positif. Un réseau contient le vecteur nul de $\mathbb{R}^d$ et il est stable par addition et soustraction.

Définition 1.18 (Un réseau [Reg08])**.** *Étant donné un ensemble de n vecteurs linéairement indépendants $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n \in \mathbb{R}^d$, le réseau généré par ces vecteurs est défini par l'ensemble*

$$\mathcal{L}(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n) = \left\{ \sum_i z_i \boldsymbol{b}_i \text{ avec } z_i \in \mathbb{Z} \right\}.$$

Les vecteurs $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n$ représentent une base du réseau. D'une manière équivalente, si on considère $\boldsymbol{B}$ la matrice $d \times n$, ayant les vecteurs $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n \in \mathbb{R}^d$ comme colonnes, alors le réseau généré par $\boldsymbol{B}$ est défini par

$$\mathcal{L}(\boldsymbol{B}) = \mathcal{L}(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n) = \{\boldsymbol{Bz} : \boldsymbol{z} \in \mathbb{Z}^n\}.$$

Les entiers n et d désignent respectivement le rang et la dimension du réseau. Lorsque $n = d$, le réseau est dit de rang plein. Dans ce manuscrit, nous nous intéressons aux réseaux de rang plein et nous notons un réseau par $\mathcal{L}$ tout court sans préciser sa base.

Orthogonalisation de Gram-Schmidt. Soient n vecteurs linéairement indépendants $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n \in \mathbb{R}^n$. Supposons que $\boldsymbol{S}$ est la matrice $n \times n$ ayant les vecteurs $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n$ comme colonnes. Nous notons par $\tilde{\boldsymbol{S}} = \tilde{\boldsymbol{s}}_1 \| \ldots \| \tilde{\boldsymbol{s}}_n$ la matrice obtenue après l'orthogonalisation de Gram-Schmidt appliquée sur la matrice $\boldsymbol{S}$. Autrement dit, nous avons :

$$\tilde{\boldsymbol{s}}_1 = \boldsymbol{s}_1 \text{ et } \tilde{\boldsymbol{s}}_i = \boldsymbol{s}_i - \sum_{j=1}^{i-1} \frac{\langle \boldsymbol{s}_i, \tilde{\boldsymbol{s}}_j \rangle}{\langle \tilde{\boldsymbol{s}}_j, \tilde{\boldsymbol{s}}_j \rangle} \tilde{\boldsymbol{s}}_j, \text{ pour } i = 2, \ldots, n.$$

Où $\langle \cdot, \cdot \rangle$ désigne le produit scalaire usuel.

Définition 1.19 (Espace engendré)**.** *L'espace engendré par un réseau $\mathcal{L}(\boldsymbol{B})$ est l'espace engendré par ses vecteurs, autrement dit nous avons,*

$$vect(\mathcal{L}(\boldsymbol{B})) = vect(\boldsymbol{B}) = \{\boldsymbol{Bx} : \boldsymbol{x} \in \mathbb{R}^n\}.$$

Définition 1.20 (Minimums successifs)**.** *Soit $\mathcal{L}(\boldsymbol{B})$ un réseau de dimension n. Pour $i \in [n]$ nous définissons le i-ème minimum successif par :*

$$\lambda_i^{\mathsf{p}}(\mathcal{L}) = min\left\{r \mid dim(vect(\mathcal{L}(\boldsymbol{B}) \cap B^{\mathsf{p}}(0,r))) \geq i\right\}.$$

avec $B^{\mathsf{p}}(0,r) = \{\boldsymbol{x} \in \mathbb{R}^n : \|\boldsymbol{x}\|_{\mathsf{p}} \leq r\}$ la boule fermée de centre l'origine et de rayon r et p un entier dans $[1,\infty) \cup \{\infty\}$.
Pour $i = 1$, $\lambda_1^{\mathsf{p}}(\mathcal{L}(\boldsymbol{B}))$ est la longueur de plus petit vecteur du réseau $\mathcal{L}(\boldsymbol{B})$ pour la norme ℓ_{p}.

Un problème algorithmique est défini par une description de tous ses paramètres et une propriété que la réponse ou bien la solution doit satisfaire. Une instance d'un problème est obtenu en spécifiant des valeurs pour les paramètres et elle peut être définie par une version recherche, optimisation ou bien décisionnel. Dans ce mémoire, nous nous intéressons à la version recherche des problèmes qui consiste étant donné une entrée $x \in \{0,1\}^\star$ à trouver une réponse $y \in \{0,1\}^\star$ qui satisfait une certaine relation avec x.

Le problème de trouver le plus petit vecteur d'un réseau $\mathcal{L}$, noté par SVP (pour *Shortest Vector Problem*, en anglais) est l'un des problèmes standards les plus étudiés dans l'algorithmique des réseaux.

Définition 1.21 ($\mathsf{SVP}^{\mathsf{p}}$ - version recherche)**.** *Le problème $\mathsf{SVP}^{\mathsf{p}}$ est le suivant : étant donnée une base $\boldsymbol{B}$ d'un réseau $\mathcal{L}$, trouver un vecteur non nul $\boldsymbol{v} \in \mathcal{L}(\boldsymbol{B})$ tel que,*

$$\|\boldsymbol{v}\|_{\mathsf{p}} = \lambda_1^{\mathsf{p}}(\mathcal{L}(\boldsymbol{B})).$$

La variante approximation de SVP que nous allons donner dans la définition suivante consiste à trouver une approximation à un facteur près du plus petit vecteur. Le facteur d'approximation est donné par un paramètre $\gamma \geq 1$.

Définition 1.22 ($\mathsf{SVP}_\gamma^{\mathsf{p}}$ - version recherche)**.** *Le problème d'approximation $\mathsf{SVP}_\gamma^{\mathsf{p}}$ est le suivant : étant donné une base $\boldsymbol{B}$ d'un réseau $\mathcal{L}$ et un réel γ, trouver un vecteur non nul $\boldsymbol{v} \in \mathcal{L}(\boldsymbol{B})$ tel que, pour tout vecteur $\boldsymbol{x}$ dans $\mathcal{L}$*

$$\|\boldsymbol{v}\|_{\mathsf{p}} \leq \gamma \cdot \|\boldsymbol{x}\|_{\mathsf{p}}.$$

Nous donnons aussi des définitions du problème des plus petits vecteurs indépendants, noté par SIVP (pour *Shortest Independant Vectors Problem*, en anglais).

Définition 1.23 ($\mathsf{SIVP}^{\mathsf{p}}$ - version recherche)**.** *Le problème $\mathsf{SIVP}^{\mathsf{p}}$ est le suivant : étant donnée une base $\boldsymbol{B}$ d'un réseau $\mathcal{L}$ de dimension n, trouver un ensemble $\boldsymbol{S} \subset \mathcal{L}(\boldsymbol{B})$ de n vecteurs linéairement indépendants tel que $\|\boldsymbol{S}\|_{\mathsf{p}} = \lambda_n^{\mathsf{p}}(\mathcal{L}(\boldsymbol{B}))$.*

La variante approximation de ce problème est donnée dans la définition suivante.

Définition 1.24 ($\mathsf{SIVP}^{\mathsf{p}}_{\gamma}$ - version recherche)**.** *Le problème d'approximation* $\mathsf{SIVP}^{\mathsf{p}}_{\gamma}$ *est le suivant : étant donné une base* $\boldsymbol{B}$ *d'un réseau* $\mathcal{L}$ *de dimension* n *et un réel* γ*, trouver un ensemble* $\boldsymbol{S} \subset \mathcal{L}(\boldsymbol{B})$ *de* n *vecteurs linéairement indépendants tel que* $\|\boldsymbol{S}\|_{\mathsf{p}} \leq \gamma \cdot \lambda^{\mathsf{p}}_n(\mathcal{L}(\boldsymbol{B}))$.

1.5.1.1 Complexité des problèmes des réseaux

Le premier résultat sur la complexité du problème du plus petit vecteur (SVP) est celui de van Emde Boas [vEB81], qui considère la version exacte du problème. Il a montré que SVP en norme ℓ_∞ est NP-difficile et il a conjecturé la NP difficulté de SVP pour la norme ℓ_2. Prouver le résultat en norme ℓ_2 est resté un problème ouvert pendant seize ans. C'est en 1998, que Ajtai [Ajt98] a prouvé ce résultat. Dans le même papier, Ajtai a démontré qu'approximer SVP avec des facteurs qui sont petits $\gamma = 1 + o(1)$ est NP-difficile. En 2004, Khot [Kho04] a prouvé que SVP est NP-difficile à approximer avec n'importe quelle constante. D'autre part, Blömer et Seifert [BS99] ont montré que le problème SIVP est NP-difficile à approximer avec n'importe quel facteur constant.

Aujourd'hui les meilleurs algorithmes pour la résolution des problèmes SVP_γ et SIVP_γ avec un facteur d'approximation $\gamma \leq \mathsf{poly}(n)$ sont exponentiels. Le meilleur algorithme déterministe pour ces problèmes est celui de Kannan [Kan83] de complexité $n^{O(n)}$ en temps. D'autres algorithmes proposés par Micciancio et Voulgaris [MV10a, MV10b] sont exponentiels en temps et en mémoire.

Gaussiennes sur les réseaux

Soient $\boldsymbol{x}$ et $\boldsymbol{c}$ deux vecteurs de $\mathbb{R}^n$. Pour tout $s > 0$, nous considérons $\rho_{s,\boldsymbol{c}}(\boldsymbol{x}) = e^{-\pi \frac{\|(\boldsymbol{x}-\boldsymbol{c})\|^2}{s^2}}$ la fonction Gaussienne centrée en $\boldsymbol{c}$ et de variance s. Nous étendons la définition de $\rho_{s,\boldsymbol{c}}$ aux ensembles dénombrables $A \subset \mathbb{R}^n$ en prenant $\rho_{s,\boldsymbol{c}}(A) = \sum_{\boldsymbol{x}\in A} \rho_{s,\boldsymbol{c}}(\boldsymbol{x})$.

Définition 1.25 (Distribution Gaussienne discrète sur un réseau [MR07])**.** *Pour tous réseau* $\mathcal{L}$*,* $\boldsymbol{c} \in \mathbb{R}^n$ *et* $s > 0$*. Nous définissons la distribution de probabilités discrète sur* $\mathcal{L}$ *par*

$$\forall \boldsymbol{x} \in \mathcal{L}, \mathsf{D}_{\mathcal{L},s,\boldsymbol{c}}(\boldsymbol{x}) = \frac{\rho_{s,\boldsymbol{c}}(\boldsymbol{x})}{\rho_{s,\boldsymbol{c}}(\mathcal{L})}.$$

Lorsque $\boldsymbol{c}$ *n'est pas spécifié, nous supposons que c'est l'origine.*

1.5.1.2 Réduction de sécurité

Définition 1.26 (Réseaux q-aires)**.** *Soient* q, m, n *des entiers tels que* $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$*, on considère deux types de réseau définis par la matrice* $\boldsymbol{A}$ *:*

$$\mathcal{L}_q(\boldsymbol{A}) = \{\boldsymbol{y} \in \mathbb{Z}^m : \boldsymbol{y} = \boldsymbol{A}^T\boldsymbol{s} \bmod q \text{ et } \boldsymbol{s} \in \mathbb{Z}^n\}$$

$$\mathcal{L}_q^{\perp}(\boldsymbol{A}) = \{\boldsymbol{y} \in \mathbb{Z}^m : \boldsymbol{A}\boldsymbol{y} = 0 \bmod q\}$$

Définition 1.27 ($\mathsf{SIS}_{n,q,m,\beta}$ - version recherche, [MR07]). *Le problème de la plus petite solution entière* SIS *(de l'anglais Small Integer Solution Problem) est : étant donnés des entiers n, q, m, une matrice $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$ et un réel β, trouver un vecteur $\boldsymbol{z} \in \mathbb{Z}^m \setminus \{0\}$ tel que $\boldsymbol{A}\boldsymbol{z} = 0 \bmod q$ et $\|\boldsymbol{z}\|_2 \leq \beta$.*

Micciancio et Regev [MR07] ont défini une variante de ce problème, il s'agit de trouver la plus petite solution (un vecteur $\boldsymbol{z}$) d'un système d'équations défini par $\boldsymbol{A}\boldsymbol{z} = \boldsymbol{u} \bmod q$ (avec $\boldsymbol{A}$ et $\boldsymbol{u}$ choisis d'une manière uniformément aléatoire). Nous notons ce problème par ISIS (pour *Inhomogeneous Small Integer Solution Problem*, en anglais).

Définition 1.28 ($\mathsf{ISIS}_{n,q,m,\beta}$ - version recherche, [MR07]). *Le problème* ISIS *est : étant donnés des entiers n, q, m, une matrice $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$, un vecteur $\boldsymbol{u} \in \mathbb{Z}_q^n$ et un réel β, trouver un vecteur $\boldsymbol{z} \in \mathbb{Z}^m$ tel que $\boldsymbol{A}\boldsymbol{z} = \boldsymbol{u} \bmod q$ et $\|\boldsymbol{z}\|_2 \leq \beta$.*

Le lemme suivant donne une condition suffisante sous laquelle les instances du problème SIS possèdent des solutions.

Lemme 1.2 (Lemme 5.2 [MR07]). *Pour tous q, $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$ et $\beta \geq \sqrt{m}q^{n/m}$, une instance $(\boldsymbol{A}, \beta)$ du problème* SIS *admet une solution, c'est-à-dire qu'il existe $\boldsymbol{z} \in \mathbb{Z}^m \setminus \{0\}$ tel que $\boldsymbol{A}\boldsymbol{z} = 0 \bmod q$ et $\|\boldsymbol{z}\|_2 \leq \beta$.*

Réduction pire cas/cas moyen de Gentry, Peikert et Vaikuntanathan

Dans son célèbre article intitulé *Generating hard instances of lattice problems* [Ajt96], Ajtai a montré que s'il existe un algorithme probabiliste polynomial qui résout des instances choisies d'une manière uniformément aléatoire du problème SIS (pour des paramètres adéquats), alors il existe un algorithme probabiliste polynomial qui résout les instances du problème SIVP_γ pour tout réseau de dimension n avec $\gamma = \mathsf{poly}(n)$. Autrement dit, il existe une réduction polynomiale probabiliste de la résolution de SIVP_γ dans le pire cas pour $\gamma = \mathsf{poly}(n)$ à la résolution de SIS dans le cas moyen (pour des paramètres adéquats).

Remarque 1.3. *Ajtai [Ajt96], a utilisé les réseaux q-aires (voir le Définition 1.26) pour montrer sa réduction. Intuitivement, le problème* SIS *défini avec les paramètres n, q, m, β est équivalent au problème de trouver le plus petit vecteur* SVP *dans le réseau $\mathcal{L}_q^{\perp}(\boldsymbol{A})$ où $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$.*

L'amélioration du facteur d'approximation a fait l'objet de plusieurs travaux [CN97, Mic04, MR07, GPV08, MP13]. Dans le théorème suivant nous rappelons la réduction de Gentry, Peikert et Vaikuntanathan [GPV08], que nous allons utiliser dans la suite.

Théorème 1.1 ([GPV08]). *Soient n, m, q des entiers tels que $m \geq 2n \log q$, $q \geq \beta \cdot \omega(\sqrt{n \log n})$ avec β un réel tel que $\beta \leq \mathsf{poly}(n)$. Il existe une réduction polynomiale probabiliste de la résolution de SIVP_γ^2 pour $\gamma = \tilde{O}(\beta\sqrt{n})$ dans le pire cas à la résolution de $\mathsf{SIS}_{n,q,m,\beta}$ ou bien $\mathsf{ISIS}_{n,q,m,\beta}$ dans le cas moyen.*

1.5.1.3 Une famille de fonctions de hachage basée sur les réseaux

Définition 1.29. *Soient n un paramètre de sécurité et q et m deux entiers qui dépendent de n. Soit $D_n \subset \mathbb{Z}^m$, la famille de fonctions de hachage $\mathcal{H}(\mathbb{Z}_q^n, D_n, m)$ est l'ensemble des fonctions $\{h_{\boldsymbol{A}} : \boldsymbol{A} \in \mathbb{Z}_q^{n \times m}\}$ tel que pour tout $\boldsymbol{x} \in D_n$, $h_{\boldsymbol{A}}(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x} \bmod q$.*

Dans [GGH96], Goldreich, Goldwasser et Halevi ont montré que trouver des collisions sur une fonction $h_{\boldsymbol{A}} \in \mathcal{H}(\mathbb{Z}_q^n, D_n, m)$ où $D_n = \{0,1\}^m$ implique la résolution du problème SIS. En effet, supposons que $(\boldsymbol{x}, \boldsymbol{x}')$ est une collision sur $h_{\boldsymbol{A}}$, c'est-à-dire que $h_{\boldsymbol{A}}(\boldsymbol{x}) = h_{\boldsymbol{A}}(\boldsymbol{x}')$. Par conséquent, nous avons que $\boldsymbol{z} = \boldsymbol{x} - \boldsymbol{x}'$ est une solution pour le problème $\mathsf{SIS}_{n,q,m,\sqrt{m}}$ puisque $\|\boldsymbol{z}\|_2 \leq \sqrt{m}$ et $\boldsymbol{A}\boldsymbol{z} = 0 \bmod q$. De la même façon on peut montrer que la résistance à la préimage repose sur la difficulté du problème $\mathsf{ISIS}_{n,q,m,\sqrt{m}}$.

Typiquement on peut choisir $m = O(n \log q)$, pour obtenir $q = \tilde{O}(n)$. Par conséquent, d'après le Théorème 1.1, nous avons que la sécurité de la famille de fonctions de hachage $\mathcal{H}(\mathbb{Z}_q^n, D_n, m)$ repose sur la difficulté du problème SIVP_γ^2 avec $\gamma = \tilde{O}(n)$.

1.5.1.4 Schéma d'engagement de Kawachi et al.

Dans [KTX08], Kawachi et al. ont introduit un schéma d'engagement dont la sécurité est basée sur le problème SIS (voir Définition 1.27). Le schéma est défini à partir de la famille de fonctions de hachage donnée dans la section 1.5.1.3. Une fonction de hachage $h_{\boldsymbol{A}}$ de cette famille est définie par une matrice aléatoire $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$, où n est un paramètre de sécurité et m, q sont deux entiers qui dépendent de n.

L'idée de Kawachi et al. consiste à calculer un haché de la concaténation du message à engager et de la valeur aléatoire nécessaire pour la vérification de l'engagement. D'une manière plus précise, soient un message μ tel que $\mu \in \{0,1\}^{m/2}$ et un vecteur aléatoire $\boldsymbol{r}$ tel que $\boldsymbol{r} \in \{0,1\}^{m/2}$, en notant $\boldsymbol{x} = \mu \| \boldsymbol{r} \in \{0,1\}^m$ on définit $\mathsf{COM}_{\boldsymbol{A}}(\mu, \boldsymbol{r})$ par :

$$\mathsf{COM}_{\boldsymbol{A}}(\mu, \boldsymbol{r}) = h_{\boldsymbol{A}}(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x} \bmod q.$$

Plus formellement, le schéma de Kawachi et al. se présente sous la forme suivante :

$\mathsf{Init}(1^n)$: Étant donné un paramètre de sécurité 1^n, soient q, m deux entiers tels que $q = \mathsf{poly}(n)$ et $m \geq 10n \log q$. L'algorithme choisit d'une manière uniformément aléatoire une matrice $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$ et renvoie $\mathsf{param} = \boldsymbol{A}$.

$\mathsf{Commit}(\boldsymbol{A}, \mu)$: Étant donnés la matrice $\boldsymbol{A}$ et le message $\mu \in \{0,1\}^{m/2}$, l'algorithme choisit aléatoirement un vecteur $\boldsymbol{r} \in \{0,1\}^{m/2}$. Ensuite, il calcule un engagement $\boldsymbol{t}$ sur μ :

$$\boldsymbol{t} \leftarrow \mathsf{COM}_{\boldsymbol{A}}(\mu, \boldsymbol{r}).$$

L'algorithme renvoie $(c, r) = (\boldsymbol{t}, \boldsymbol{r})$.

$\mathsf{Verify}(\boldsymbol{A}, \mu, \boldsymbol{t}, \boldsymbol{r})$: L'algorithme vérifie que $\mu, \boldsymbol{r} \in \{0,1\}^{m/2}$ et $\mathsf{COM}_{\boldsymbol{A}}(\mu, \boldsymbol{r}) = \boldsymbol{t}$. Il renvoie 1 si les vérifications sont valides, sinon il renvoie 0.

La sécurité du schéma est résumée dans le lemme suivant :

Lemme 1.3. *(Lemme 3.1 dans [KTX08]) Soient n un paramètre de sécurité et q, m deux entiers tels que $q = \mathsf{poly}(n)$ et $m \geq 10n \log q$. Si le problème $\mathsf{SIS}_{n,q,m,\sqrt{m}}$ est difficile dans le cas moyen, alors le schéma d'engagement satisfait les deux propriétés de sécurité d'un schéma d'engagement, celle d'indistinguabilité et celle d'engagement.*

Remarque 1.4. *Il est démontré dans [KTX08] que le domaine du message μ peut être étendu à $\{0,1\}^\star$ au lieu de $\{0,1\}^{m/2}$.*

1.5.2 Les réseaux idéaux

Cette section est divisée en deux parties. La première est consacrée à rappeler ce que c'est un réseau idéal et à donner des définitions et des notions liées à ce type particulier de réseaux. Dans la deuxième partie nous nous intéressons à une famille de fonctions de hachage introduite par Lyubashevsky et Micciancio [LM06]. Nous utilisons les définitions et les résultats tels qu'ils sont donnés dans le manuscrit de thèse de Lyubashevsky [Lyu08b].

1.5.2.1 Définitions et problème difficile

Soit $\mathbb{Z}[x]$ l'ensemble des polynômes à coefficients entiers. Un polynôme $\mathbf{f}$ de degré n dans $\mathbb{Z}[x]$ est de la forme $\mathbf{f}(x) = a_0 + a_1 x + \ldots + a_{n-1}x^{n-1} + a_n x^n$, où $\forall i, a_i \in \mathbb{Z}$. Un polynôme est unitaire si le coefficient non nul avec le plus grand degré est égal à un. Un polynôme de degré n dans $\mathbb{Z}[x]$ est irréductible si on ne peut pas l'écrire comme produit de deux polynômes de degré inférieur à n dans $\mathbb{Z}[x]$.

Dans la suite, nous nous intéressons aux anneaux de type $\mathbb{Z}[x]/\langle \mathbf{f} \rangle$, où $\mathbf{f}$ est un polynôme unitaire. Lorsque $\mathbf{f}$ est unitaire de degré n dans $\mathbb{Z}[x]$, toutes les classes d'équivalence $(\mathbf{g} + \langle \mathbf{f} \rangle) \in \mathbb{Z}[x]/\langle \mathbf{f} \rangle$ ont un unique représentant $\mathbf{g}' \in (\mathbf{g} + \langle \mathbf{f} \rangle)$ de degré strictement inférieur à n, on le notera par $\mathbf{g} \bmod \mathbf{f}$.

Soient $\mathbf{a}, \mathbf{b}$ deux polynômes dans $\mathbb{R}[x]$ de degré n et $\mathbf{f}$ un polynôme unitaire dans $\mathbb{Z}[x]$. Le paramètre $\theta(\mathbf{f})$ que nous allons présenter dans la suite permet d'évaluer le maximum que peut valoir $\|\mathbf{ab} \bmod \mathbf{f}\|_\infty$ pour deux polynômes $\mathbf{a}$ et $\mathbf{b}$ dans $\mathbb{Z}[x]$. Comme il est mentionné dans [Lyu08b], la multiplication de deux polynômes modulo $\mathbf{f}$ dépend essentiellement de la multiplication d'un polynôme par une puissance de x.

Définition 1.30 (Le paramètre $\theta(\mathbf{f})$ [Lyu08b]). *Soit $\mathbf{f} \in \mathbb{Z}[x]$ un polynôme unitaire.*

$$\theta(\mathbf{f}) = min\{j : \forall \mathbf{a} \in \mathbb{R}[x] \text{ de degré } < n \text{ et } 0 \leq i \leq n-1, \|\mathbf{a}x^i \bmod \mathbf{f}\|_\infty \leq j\|\mathbf{a}\|_\infty\}.$$

Lemme 1.4 (Lemme 2.6 dans [Lyu08b]). *Si $\mathbf{f} = x^n + 1$, alors $\theta(\mathbf{f}) = 1$.*

Réseau idéal (*ideal lattice*). Suivant [Lyu08b], soit $\mathbf{f} \in \mathbb{Z}[x]$ un polynôme unitaire de degré n, nous considérons l'anneau $\mathbb{Z}[x]/\langle \mathbf{f} \rangle$. Un réseau idéal est identifié grâce à une correspondance entre un réseau $\mathcal{L} \subset \mathbb{Z}^n$ et un idéal $I \subset \mathbb{Z}[x]/\langle \mathbf{f} \rangle$. On dit qu'un réseau $\mathcal{L}$ et un idéal I se correspondent l'un à l'autre si tout vecteur $(v_0, \cdots, v_{n-1})$ est dans $\mathcal{L}$ si seulement si le polynôme $v^{(0)} + v^{(1)}x + \cdots + v^{(n-1)}x^{n-1}$ est dans I, où $v_i = v^{(i)}$ pour tout i. Par conséquent, un réseau idéal $\mathcal{L}(I)$ est un réseau qui correspond à un idéal $I \subset \mathbb{Z}[x]/\langle \mathbf{f} \rangle$.

Lemme 1.5 (Lemme 2.12 dans [Lyu08b]). *Soit $\mathbf{f} \in \mathbb{Z}[x]$ un polynôme irréductible de degré n, tout idéal I de $\mathbb{Z}[x]/\langle \mathbf{f} \rangle$ est isomorphe à un réseau $\mathcal{L}$ de rang plein dans $\mathbb{Z}^n$.*

Soient $\mathcal{L}(I)$ un réseau idéal, et p un entier dans $[1, \infty) \cup \{\infty\}$, les minima successifs $\lambda_i^{\mathsf{p}}(\mathcal{L}(I))$ sont définis de la même façon que les réseaux classiques (voir la Définition 1.20). Le problème d'approximer le plus petit vecteur d'un réseau idéal $\mathcal{L}(I)$ est donné dans la définition suivante.

Définition 1.31 ($\mathbf{f}$-$\mathsf{SVP}_\gamma^\infty$ - version recherche). *Étant donnés $\gamma \geq 1$, et un réseau idéal $\mathcal{L}(I)$ qui correspond à un idéal $I \subseteq \mathbb{Z}[x]/\langle \mathbf{f} \rangle$, où $\mathbf{f} \in \mathbb{Z}[x]$ est un polynôme unitaire. Le but est de trouver un élément $\mathbf{v} \in \mathcal{L}(I)$ tel que $\|\mathbf{v}\|_\infty \leq \gamma \cdot \lambda_1^\infty(\mathcal{L}(I))$.*

Il est conjecturé que résoudre le problème $\mathbf{f}$-$\mathsf{SVP}_\gamma^\infty$ est aussi difficile que la résolution du problème $\mathsf{SVP}_\gamma^\infty$. Nous avons la conjecture suivante.

Théorème 1.2 (Conjecture 2.1 dans [Lyu08b]). *Pour tout polynôme unitaire $\mathbf{f}$ de degré n et pour toute constante c, résoudre le problème $\mathbf{f}$-$\mathsf{SVP}_\gamma^\infty$ pour $\gamma = O(n^c)$ nécessite une complexité en temps de $2^{\Omega(n)}$.*

1.5.2.2 Famille de fonctions de hachage basée sur les réseaux idéaux

Soit n un entier, nous considérons l'anneau $\mathcal{D} = \mathbb{Z}_p[x]/\langle \mathbf{f} \rangle$ où $p = \mathsf{poly}(n)$ est un entier et $\mathbf{f}$ est un polynôme de degré n irréductible dans $\mathbb{Z}[x]$. Les éléments de cet

anneau seront représentés par des polynômes de degré $n-1$ avec des coefficients dans $\{-(p-1)/2, \cdots, (p-1)/2\}$.

Dans [LM06], Lyubashevsky et Micciancio ont introduit une famille de fonctions de hachage résistante aux collisions avec une sécurité basée sur le problème $\mathbf{f}$-SVP$_\gamma^\infty$ (voir Définition 1.31). Nous donnons une description de cette famille dans la définition suivante.

Définition 1.32. *Soient m un entier positif et $D_\times \subseteq \mathcal{D}$. La famille de fonctions de hachage $\mathcal{H}(\mathcal{D}, D_\times, m)$ est une collection de fonctions $\{h_{\hat{\mathbf{a}}} : \hat{\mathbf{a}} \in \mathcal{D}^m\}$ telle que pour tout $\hat{\mathbf{z}} \in D_\times^m$, $h_{\hat{\mathbf{a}}}(\hat{\mathbf{z}}) = \hat{\mathbf{a}} \cdot \hat{\mathbf{z}} = \sum_{i=1}^m \mathbf{a}_i \mathbf{z}_i$, avec $\hat{\mathbf{a}} = (\mathbf{a}_1, \cdots, \mathbf{a}_m)$ et $\hat{\mathbf{z}} = (\hat{\mathbf{z}}_1, \cdots, \hat{\mathbf{z}}_m)$. Toutes les opérations sont effectuées dans l'anneau $\mathcal{D}$.*

Une fonction de hachage $h_{\hat{\mathbf{a}}} \in \mathcal{H}(\mathcal{D}, D_\times, m)$ satisfait les propriétés suivantes pour tous $\hat{\mathbf{y}}, \hat{\mathbf{z}} \in \mathcal{D}^m$ et $\mathbf{c} \in \mathcal{D}$:

$$h_{\hat{\mathbf{a}}}(\hat{\mathbf{y}} + \hat{\mathbf{z}}) = h_{\hat{\mathbf{a}}}(\hat{\mathbf{y}}) + h_{\hat{\mathbf{a}}}(\hat{\mathbf{z}}) \tag{1.1}$$

$$h_{\hat{\mathbf{a}}}(\hat{\mathbf{y}}\mathbf{c}) = h_{\hat{\mathbf{a}}}(\hat{\mathbf{y}})\mathbf{c} \tag{1.2}$$

Remarque 1.5. *Pour simplifier les notations, nous notons par h une fonction de hachage $h_{\hat{\mathbf{a}}} \in \mathcal{H}(\mathcal{D}, D_\times, m)$.*

En outre, lorsque l'espace de départ est limité à un ensemble bien choisi $D_\times^m \subseteq \mathcal{D}^m$, la famille de fonctions est résistante aux collisions. Nous introduisons le problème de trouver des collisions pour cette famille de fonctions, ensuite nous donnerons la réduction de sécurité.

Définition 1.33. *Le problème de collision noté par $\mathsf{Col}(h, D_\times)$, (avec $D_\times \subseteq \mathcal{D}$) est comme suit : Étant donnée une fonction $h \in \mathcal{H}(\mathcal{D}, D_\times, m)$, trouver deux éléments distincts $\hat{\mathbf{z}}_1, \hat{\mathbf{z}}_2 \in D_\times^m$ tels que $h(\hat{\mathbf{z}}_1) = h(\hat{\mathbf{z}}_2)$.*

Dans [LM06], il est montré que si $D_\times$ est restreint à un ensemble particulier, alors résoudre le problème $\mathsf{Col}(h, D_\times)$ est aussi difficile que résoudre le problème $\mathbf{f}$-SVP$_\gamma^\infty$ dans le pire cas (pour un certain γ).

Théorème 1.3. *Soient n un paramètre de sécurité, d un entier positif qui dépend de n, un entier $m \geq \frac{\log p}{\log 2d}$, et un premier $p \geq 4dmn^{1.5} \log n$. Soit l'anneau $\mathcal{D} = \mathbb{Z}_p[x]/\langle \mathbf{f} \rangle$, tel que $\mathbf{f}$ un polynôme irréductible dans $\mathbb{Z}[x]$ de degré n. Soient $D_\times = \{\mathbf{y} \in \mathcal{D} : \|\mathbf{y}\|_\infty \leq d\}$ et $\mathcal{H}(\mathcal{D}, D_\times, m)$, une famille de fonctions de hachage comme dans la Définition 1.32. S'il existe un algorithme en temps polynomial qui résout le problème $\mathsf{Col}(h, D_\times)$ pour $h \in \mathcal{H}(\mathcal{D}, D_\times, m)$ avec une probabilité non négligeable, alors il existe un algorithme en temps polynomial qui peut résoudre $\mathbf{f}$-SVP$_\gamma^\infty$ dans le pire cas pour $\gamma = 16\theta(\mathbf{f})^2 dmn \log^2 n$.*

1.5.3 Outils cryptographiques basés sur les réseaux

1.5.3.1 Fonctions à pré-image échantillonable

Tout d'abord, nous rappelons qu'une fonction f est dite à trappe si :
— il existe une information appelée trappe, qui permet pour tout y image de f, d'obtenir un x tel que $f(x) = y$;
— il est difficile pour tout y choisi uniformément dans l'image de f d'obtenir un x tel que $f(x) = y$ sans la trappe.

En 2008, Gentry, Peikert et Vaikuntanathan [GPV08] ont introduit la notion de fonction à pré-image échantillonable (en anglais, *preimage-samplable function*). Nous allons rappeler cette notion en suivant les définitions données dans [GPV08]. Une famille de fonctions à pré-image échantillonable est définie par le quadruplet d'algorithmes (TrapGen, Eval, SampleDom, SamplePre).

TrapGen(1^n) : est un algorithme probabiliste polynomial qui prend en entrée le paramètre de sécurité 1^n et retourne (a, t), avec a une description d'une fonction $f_a : D_n \to R_n$ et t une trappe pour f_a (D_n, R_n sont deux domaines qui dépendent du paramètre de sécurité).

Eval(a, x) : est un algorithme qui prend en entrée a et $x \in D_n$ et retourne $y = f_a(x)$.

SampleDom(1^n) : est un algorithme probabiliste polynomial qui prend en entrée le paramètre de sécurité 1^n et retourne $x \in D_n$, qui suit une certaine distribution sur D_n.

SamplePre(t, y) : est un algorithme probabiliste polynomial qui prend en entrée une trappe t et $y \in R_n$ et retourne x qui suit une certaine distribution sur D_n tel que $f_a(x) = y$.

Soient n, q, m, s des paramètres. D'une manière informelle, une fonction qui appartient à la famille de fonctions à pré-image échantillonable proposée par Gentry et al. dans [GPV08] est définie par une matrice $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$, notée $f_{\boldsymbol{A}}$ telle que : $f_{\boldsymbol{A}} : D_n \to R_n$ avec $f_{\boldsymbol{A}}(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x} \bmod q$ où $D_n = \{\boldsymbol{x} \in \mathbb{Z}^m : 0 < \|\boldsymbol{x}\|_2 \leq s \cdot \sqrt{m}\}$ et $R_n = \mathbb{Z}_q^n$. En suivant [GPV08], nous donnons une définition formelle de cette famille, que nous notons par PSF (en anglais, *Preimage-Samplable Function*).

Paramètres : soient n le paramètre de sécurité et q, m deux entiers tels que $q = \mathsf{poly}(n)$ et $m \geq 5n \log q$.

TrapGen(1^n) : l'algorithme retourne deux matrices $(\boldsymbol{A}, \boldsymbol{S}) \in \mathbb{Z}_q^{n \times m} \times \mathbb{Z}^{m \times m}$ telles que $\boldsymbol{S}$ est une base de $\mathcal{L}_q^{\perp}(\boldsymbol{A})$ et $\|\tilde{\boldsymbol{S}}\|_2 \leq O(\sqrt{n \log q})$, où $\tilde{\boldsymbol{S}}$ est la matrice d'orthogonalisation de Gram-Schmidt de la matrice $\boldsymbol{S}$. Le fonction à trappe $f_{\boldsymbol{A}}$ est définie par $f_{\boldsymbol{A}}(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x} \bmod q$. La matrice $\boldsymbol{S}$ est une trappe pour la fonction $f_{\boldsymbol{A}}$.

$\mathsf{Eval}(\boldsymbol{A}, \boldsymbol{x})$: étant donnés la matrice $\boldsymbol{A}$ et un vecteur $\boldsymbol{x} \in \mathbb{Z}^m$, l'algorithme retourne $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} \bmod q$.

$\mathsf{SampleDom}(m, s)$: l'algorithme prend en paramètre m et un entier $s \geq \|\tilde{\boldsymbol{S}}\| \cdot \omega(\sqrt{\log m})$. Il retourne un vecteur $\boldsymbol{x}$ choisi à partir de la distribution gaussienne $\mathsf{D}_{\mathbb{Z}^m,s}$, en particulier $\boldsymbol{x} \in D_n$ avec une probabilité proche de 1.

$\mathsf{SamplePre}(\boldsymbol{A}, \boldsymbol{S}, \boldsymbol{y}, s)$: l'algorithme retourne un vecteur $\boldsymbol{x}$ ayant une distribution statistiquement proche de la distribution $\mathsf{D}_{\mathbb{Z}^m,s}$ tel que $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y} \bmod q$, en particulier $\boldsymbol{x} \in D_n$ avec un probabilité proche de 1.

Nous résumons la sécurité de cette famille de fonctions dans la proposition suivante.

Proposition 1.6. *Il est démontré dans [GPV08] que la famille* PSF *satisfait les propriétés suivantes :*

1. *Toute fonction $f_{\boldsymbol{A}} \in$ PSF est une fonction à sens unique et résistante aux collisions sous l'hypothèse que le problème* SIS *est difficile dans le cas moyen.*
2. *Pour tout $(\boldsymbol{A}, \boldsymbol{S}) \leftarrow \mathsf{TrapGen}(1^n)$, la matrice $\boldsymbol{A}$ a une distribution statistiquement proche de la distribution uniforme sur $\mathbb{Z}_q^{n \times m}$.*
3. *Pour tout $(\boldsymbol{A}, \boldsymbol{S}) \leftarrow \mathsf{TrapGen}(1^n)$, nous avons que $\Delta(X, Y) \leq \mathsf{negl}(n)$, où $X = \mathsf{Eval}(\boldsymbol{A}, \mathsf{SampleDom}(m, s))$ et Y est la distribution uniforme sur $\mathbb{Z}_q^n$.*
4. *Pour tout $(\boldsymbol{A}, \boldsymbol{S}) \leftarrow \mathsf{TrapGen}(1^n)$, nous avons que $\Delta(X, Y) \leq \mathsf{negl}(n)$, où X est la variable aléatoire égale à $(\boldsymbol{x}, \boldsymbol{y})$ telle que $\boldsymbol{x} \leftarrow \mathsf{SampleDom}(m, s)$ et $\boldsymbol{y} \leftarrow \mathsf{Eval}(\boldsymbol{A}, \boldsymbol{x})$ et Y est la variable aléatoire égale à $(\boldsymbol{x}, \boldsymbol{y})$ telle que $\boldsymbol{y}$ est choisi d'une manière uniformément aléatoire dans $\mathbb{Z}_q^n$ et $\boldsymbol{x} \leftarrow \mathsf{SamplePre}(\boldsymbol{A}, \boldsymbol{S}, \boldsymbol{y}, s)$.*

1.5.3.2 Échantillonnage généralisé de pré-image

Dans [CHK09], Cash, Hofheinz et Kiltz ont proposé une généralisation de l'algorithme SamplePre présenté dans la section 1.5.3.1. Soient n, q, m des entiers comme dans la section 1.5.3.1 et un entier $k > 0$. Soit $\boldsymbol{A} \in \mathbb{Z}_q^{n \times km}$ telle que $\boldsymbol{A} = [\boldsymbol{A}_1 \| \cdots \| \boldsymbol{A}_k]$ où $\boldsymbol{A}_i \in \boldsymbol{A}_q^{n \times m}$ pour $i \in [k]$. Soit $\boldsymbol{A}_N = [\boldsymbol{A}_{i_1} \| \cdots \| \boldsymbol{A}_{i_j}]$, avec $\{i_1, \cdots, i_j\} \subset [k]$. Le nouveau algorithme prend en entrée $\boldsymbol{A}$, $\boldsymbol{A}_N$, une base $\boldsymbol{S}_N$ de $\mathcal{L}_q^{\perp}(\boldsymbol{A}_N)$, un vecteur $\boldsymbol{y} \in \mathbb{Z}_q^n$, et un entier $s \geq \|\tilde{\boldsymbol{S}}_N\|_2 \cdot \omega(\sqrt{\log km})$. Il retourne un vecteur $\boldsymbol{x}$ choisi à partir de la distribution gaussienne sur $\mathsf{D}_{\mathbb{Z}^{km},s}$ tel que $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y} \bmod q$. En particulier $0 < \|\boldsymbol{x}\|_2 \leq s \cdot \sqrt{km}$ avec une probabilité proche de 1. Nous utilisons la notation

$$\boldsymbol{x} \leftarrow \mathsf{GenSamplePre}(\boldsymbol{A}, \boldsymbol{A}_N, \boldsymbol{S}_N, \boldsymbol{y}, s).$$

Proposition 1.7 (Théorème 3.4 dans [CHK09])**.** *La sortie de l'algorithme* GenSamplePre *est à une distance statistique négligeable de la distribution $\mathsf{D}_{\mathbb{Z}^{km},s}$.*

1.5.3.3 Fonctions à trappes de cercle

Dans [BK10], Brakerski et Kalai ont introduit la notion des fonctions à trappe de cercle. Il s'agit d'une généralisation du principe des fonctions à trappe classiques. D'une manière informelle, une famille de fonctions à trappe de cercle est une collection de fonctions $\mathcal{T} = \{\mathcal{T}_k\}_{k\in\mathbb{N}}$ avec $\mathcal{T}_k = \{f : X_k \to Y_k\}$, où $X = \{X_k\}_{k\in\mathbb{N}}$ est une collection d'ensembles bien définie, et $Y = \{Y_k\}_{k\in\mathbb{N}}$ est une collection de groupes commutatifs telle que :

— étant donnés $f_1, \cdots, f_t \leftarrow X_k$ et $y \leftarrow Y_k$, il doit être difficile de trouver $x_1, \cdots, x_t \in X_k$ tels que $\sum_{i=1}^{t} f_i(x_i) = y$.

— étant donnés une trappe de n'importe quelle fonction f_i, et $y \in Y_k$ quelconque, il doit être possible de calculer $x_1, \cdots, x_t \in X_k$ tels que $\sum_{i=1}^{t} f_i(x_i) = y$. De plus, il doit être impossible à quelqu'un qui voit $x_1, \cdots, x_t$ de deviner quelle trappe a été utilisée pour les générer.

Les auteurs de [BK10], ont montré comment construire une famille de fonctions à trappe qui repose sur la difficulté dans le cas moyen des problèmes SIS et ISIS. Nous rappelons dans la suite cette construction.

Fonctions à trappe de cercle. Un élément $\mathcal{T}_k$ de la famille $\mathcal{T}$ a comme paramètres :

— q un entier polynomial en le paramètre de sécurité k, $(q = \mathsf{poly}(k))$,

— un entier $m = (5+\delta)k\log q$ (pour un entier $\delta > 0$),

— un entier $s = L \cdot \omega(\sqrt{\log k})$, où $L = O(\sqrt{k\log q})$.

— $X_k = \{\boldsymbol{x} \in \mathbb{Z}_q^m : \|\boldsymbol{x}\|_2 \leq s\cdot\sqrt{m}\}$ et $Y_k = \mathbb{Z}_q^k$.

Nous avons les propriétés suivantes :

Échantillonnage : Étant donné 1^k, les fonctions de $\mathcal{T}_k$ sont choisies de deux manières différentes. La première consiste à choisir une matrice aléatoire dans $\mathbb{Z}_q^{k\times m}$, la fonction $f_{\boldsymbol{A}}$ est définie par $f_{\boldsymbol{A}}(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x} \bmod q$. La deuxième consiste à générer une fonction à trappe en utilisant l'algorithme TrapGen (voir la section 1.5.3.1) qui retourne une paire de matrices $(\boldsymbol{A}, \boldsymbol{S}) \in \mathbb{Z}_q^{k\times m} \times \mathbb{Z}_q^{m\times m}$. De même, la fonction $f_{\boldsymbol{A}}$ est définie par $f_{\boldsymbol{A}}(\mathbf{x}) = \boldsymbol{A}\boldsymbol{x} \bmod q$.

Trappe : Étant donnés $\boldsymbol{A}_1, \cdots, \boldsymbol{A}_t$, une trappe $\boldsymbol{S}_i$ de $\boldsymbol{A}_i$ et $\boldsymbol{y} \in \mathbb{Z}_q^k$, il est possible de générer $\boldsymbol{x}_j \leftarrow X_k$ pour $j \neq i$ (il suffit d'utiliser l'algorithme SampleDom). Ensuite, en utilisant l'algorithme SamplePre avec $(\boldsymbol{A}_i, \boldsymbol{S}_i, \boldsymbol{y} - \sum_{j\neq i}\boldsymbol{A}_j\boldsymbol{x}_j, s)$ comme entrée il est possible d'obtenir $\boldsymbol{x}_i \in X_k$ tel que $\sum_{j=1}^{t}\boldsymbol{A}_j\boldsymbol{x}_j = \boldsymbol{y}$. En outre, le fait d'utiliser une autre trappe $\boldsymbol{S}_j$ $(j \neq i)$ donne une distribution de $(\boldsymbol{x}_1, \cdots, \boldsymbol{x}_t)$ qui est statistiquement proche de la distribution des vecteurs générés grâce à $\boldsymbol{S}_i$.

La propriété de sens-unique est donnée dans la proposition suivante :

Proposition 1.8. *S'il existe un attaquant qui étant donnés $f_{\boldsymbol{A}_1}, \cdots, f_{\boldsymbol{A}_t} \leftarrow \mathcal{T}_k$ et $\boldsymbol{y} \in \mathbb{Z}_q^k$, peut trouver $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_t \in X_k$ tel que $\sum_{i=1}^{t} f_{\boldsymbol{A}_i}(\boldsymbol{x}_i) = \boldsymbol{y} \bmod q$. Alors, il existe un algorithme qui peut résoudre le problème $\mathsf{ISIS}_{k,q,tm,s\sqrt{tm}}$ dans*

le cas moyen.

1.6 Schémas d'identification basés sur les réseaux

Nous allons rappeler trois schémas d'identification en trois passes que nous utiliserons dans la suite de ce mémoire.

1.6.1 Un premier schéma d'identification de Lyubashevsky

Dans cette section nous rappelons le schéma d'identification en trois passes de Lyubashevsky [Lyu08a], que nous nommons LID08. Soient n le paramètre de sécurité, $m = \lceil 4n \log n \rceil$, $q = \tilde{O}(n^3)$, $t = \omega(\log n)$ et $\mathsf{SAFE} = \{1, \cdots, 5m-1\}^m$.

$((\boldsymbol{y}, \boldsymbol{A}), \boldsymbol{x}) \leftarrow \mathsf{Id{-}keygen}(1^n)$:

1. Soit $\boldsymbol{x} \leftarrow \{0,1\}^m$.
2. Soit $\boldsymbol{A} \leftarrow \mathbb{Z}_q^{n \times m}$.
3. Soit $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} \bmod q$.

$(\mathcal{P} \leftrightarrow \mathcal{V})$: Les paramètres en commun entre le prouveur et le vérificateur sont $(\boldsymbol{y}, \boldsymbol{A})$. Le prouveur possède la clé secrète $\boldsymbol{x}$ comme entrée auxiliaire.

1. — $\mathcal{P}$ choisit $\{\boldsymbol{s}_1, \cdots, \boldsymbol{s}_t\} \leftarrow \{0, 1, \cdots, 5m-1\}^m$.
   — $\mathcal{P}$ calcule $\boldsymbol{z}_i = \boldsymbol{A}\boldsymbol{s}_i \bmod q$, pour $i \in [t]$.
   — $\mathcal{P}$ envoie $\boldsymbol{z}_1, \cdots, \boldsymbol{z}_t$ à $\mathcal{V}$.
2. $\mathcal{V}$ choisit $\{b_1, \cdots, b_t\} \leftarrow \{0,1\}$ et il l'envoie à $\mathcal{P}$.
3. $\mathcal{P}$ calcule la réponse de la manière suivante : pour tout $i \in [t]$
   — Si $b_i = 1$ et $\boldsymbol{s}_i + \boldsymbol{x} \notin \mathsf{SAFE}$, alors $\boldsymbol{r}_i = \perp$.
   — Sinon $\boldsymbol{r}_i = \boldsymbol{s}_i + \boldsymbol{x}$.
   $\mathcal{P}$ envoie $\boldsymbol{r}_1, \cdots, \boldsymbol{r}_t$ à $\mathcal{V}$.

. Vérification : $\mathcal{V}$ vérifie la réponse de la manière suivante : pour tout $i \in [t]$
— Si $\|\boldsymbol{r}_i\|_2 \leq 5m^{1.5}$ et $\boldsymbol{A}\boldsymbol{r}_i \bmod q = b_i\boldsymbol{y} + \boldsymbol{z}_i$, alors $d_i = 1$.
— Sinon, $d_i = 0$.
Soit $sum = \sum_{i=1}^{t} d_i$. $\mathcal{V}$ retourne $\mathsf{desc} = 1$ si $sum \geq 0.65t$, sinon il retourne $\mathsf{desc} = 0$.

Sécurité. Ce schéma est sûr contre les attaques actives (voir Définition 1.13) si le problème SIS est difficile. Plus précisément, s'il existe un attaquant en temps polynomial $\mathcal{A}$ qui casse le protocole, alors il existe un algorithme en temps polynomial qui peut résoudre une instance aléatoire du problème SIS.

Au début du jeu le challengeur reçoit une matrice aléatoire $\boldsymbol{A} \leftarrow \mathbb{Z}_q^{n \times m}$. Ce dernier choisit comme clé secrète un vecteur $\boldsymbol{x} \leftarrow \{0,1\}^m$ et il calcule $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} \bmod q$.

Ensuite, il initialise l'attaquant en lui donnant $(\boldsymbol{y}, \boldsymbol{A})$ comme clé publique. Par conséquent le challengeur peut simuler correctement le schéma d'identification avec l'attaquant qui joue le rôle d'un vérificateur pendant la phase de l'apprentissage. En utilisant les réponses de l'attaquant pendant la phase de challenge, il est possible d'extraire une solution du problème SIS défini par la matrice $\boldsymbol{A}$. En effet, le challengeur dans le rôle d'un vérificateur répond par t challenges $\{b_i\}_{i\in[t]}$ après avoir reçu $\{\boldsymbol{z}_i\}_{i\in[t]}$ comme un engagement de la part de l'attaquant. À son tour l'attaquant répond par $\{\boldsymbol{r}_i\}_{i\in[t]}$. Le challengeur relance l'attaquant avec un nouveau challenge $\{b'_i\}_{i\in[t]}$ pour le même engagement et obtient une autre réponse $\{\boldsymbol{r}'_i\}_{i\in[t]}$. Avec un probabilité non négligeable il est possible de trouver i tel que $b_i \neq b'_i$, $\boldsymbol{A}\boldsymbol{r}_i \bmod q = b_i\boldsymbol{y} + \boldsymbol{z}_i$, et $\boldsymbol{A}\boldsymbol{r}'_i \bmod q = b'_i\boldsymbol{y} + \boldsymbol{z}_i$. Supposons que $b_i = 1$ et $b'_i = 0$, alors on obtient que

$$\boldsymbol{A}(\boldsymbol{r}_i - \boldsymbol{r}'_i) \bmod q = \boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} \bmod q.$$

Par conséquent, on obtient une solution au problème $\mathsf{SIS}_{n,q,m,\beta}$, puisqu'on a $\boldsymbol{A}(\boldsymbol{r}_i - \boldsymbol{r}'_i - \boldsymbol{x}) = 0 \bmod q$, pour un certain $\beta = \mathsf{poly}(n)$. Afin de montrer que la différence entre $\boldsymbol{r}_i - \boldsymbol{r}'_i$ et $\boldsymbol{x}$ est non nulle, Lyubashevsky utilise le fait que ce schéma satisfait la propriété de témoin indistinguable et que chaque clé publique $\boldsymbol{y}$ possède deux clés secrètes valides, c'est-à-dire qu'il existe $\boldsymbol{x}' \in \{0,1\}^m$ tel que $\boldsymbol{x}' \neq \boldsymbol{x}$ et $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}\boldsymbol{x}' = \boldsymbol{y} \bmod q$.

1.6.2 Un deuxième schéma d'identification de Lyubashevsky

Dans cette section nous rappelons le schéma d'identification en trois passes de Lyubashevsky [Lyu09], que nous nommons LID09.

Soient k le paramètre de sécurité, $\mathcal{H}(\mathcal{D}, D_\times, m)$ une famille de fonctions de hachage, comme dans la section 1.5.2.2, avec $D_\times = D_h$ et $m = 3\log n$ (voir la Table 1.1).

$((h, \mathbf{S}), \hat{\mathbf{s}}) \leftarrow \mathsf{Id{-}keygen}(1^k)$:

	Définition
n	puissance de 2 plus grande que le paramètre de sécurité k
p	premier de l'ordre de $\Theta(n^4)$
m	$3\log n$
$\mathcal{D}$	l'anneau $\mathbb{Z}_p[x]/\langle x^n + 1\rangle$
D_h	$\{\mathbf{g} \in \mathcal{D} : \|\mathbf{g}\|_\infty \leq mn^2\}$
D_y	$\{\mathbf{g} \in \mathcal{D} : \|\mathbf{g}\|_\infty \leq mn^2\}$
D_z	$\{\mathbf{g} \in \mathcal{D} : \|\mathbf{g}\|_\infty \leq mn^2 - n\}$
$D_{s,c}$	$\{\mathbf{g} \in \mathcal{D} : \|\mathbf{g}\|_\infty \leq 1\}$

TABLE 1.1 – Ensembles et paramètres (Figure 5.1 dans [Lyu08b]).

1. Soit $h \leftarrow \mathcal{H}(\mathcal{D}, D_\times, m)$.
2. Soit $\hat{\mathbf{s}} \leftarrow D_{s,c}^m$.
3. Soit $\mathbf{S} = h(\hat{\mathbf{s}})$.

$(\mathcal{P} \leftrightarrow \mathcal{V})$: Les paramètres en commun entre le prouveur et le vérificateur sont $(h, \mathbf{S})$. Le prouveur possède la clé secrète $\hat{\mathbf{s}}$ comme entrée auxiliaire.

1. $\mathcal{P}$ choisit $\hat{\mathbf{y}} \leftarrow D_y^m$, il calcule $\mathbf{Y} = h(\hat{\mathbf{y}})$ et il envoie $\mathbf{Y}$ à $\mathcal{V}$.
2. $\mathcal{V}$ choisit $\mathbf{c} \leftarrow D_{s,c}$ et il l'envoie à $\mathcal{P}$.
3. $\mathcal{P}$ calcule $\hat{\mathbf{z}} = \hat{\mathbf{s}}\mathbf{c} + \hat{\mathbf{y}}$. Si $\hat{\mathbf{z}} \notin D_z^m$, alors $\hat{\mathbf{z}} = \perp$. $\mathcal{P}$ envoie $\hat{\mathbf{z}}$ à $\mathcal{V}$.

. Vérification : $\mathcal{V}$ vérifie si :
— $\hat{\mathbf{z}} \in D_z^m$
— $h(\hat{\mathbf{z}}) = \mathbf{S}\mathbf{c} + \mathbf{Y}$
$\mathcal{V}$ retourne desc $= 1$ si les tests sont valides, sinon il retourne desc $= 0$.

Sécurité. Ce schéma est sûr contre les attaques actives (voir Définition 1.13) si le problème Col$(h, D_\times)$ est difficile pour h choisie aléatoirement dans $\mathcal{H}(\mathcal{D}, D_\times, m)$. Au début du jeu le challengeur reçoit une fonction de hachage $h \in \mathcal{H}(\mathcal{D}, D_\times, m)$. Ce dernier choisit une clé secrète $\hat{\mathbf{s}} \leftarrow D_{s,c}^m$ et obtient $\mathbf{S} = h(\hat{\mathbf{s}})$. Ensuite il initialise l'attaquant en lui donnant $(h, \mathbf{S})$ comme clé publique. Durant la phase d'apprentissage, le challengeur peut simuler parfaitement un prouveur honnête puisqu'il connait la clé secrète $\hat{\mathbf{s}}$. En utilisant les réponses de l'attaquant pendant la phase de challenge, il est possible d'extraire une collision sur la fonction h. En effet, le challengeur dans le rôle d'un vérificateur reçoit un engagement $\mathbf{Y}$, il l'enregistre et envoie un challenge $\mathbf{c} \in D_{s,c}$. Après la réception de la réponse $\hat{\mathbf{z}}$, le challengeur relance l'attaquant avec un nouveau challenge $\mathbf{c}' \in D_{s,c}$ pour le même engagement $\mathbf{Y}$ et obtient une autre réponse $\hat{\mathbf{z}}'$. En utilisant les réponses de l'attaquant et puisque le challengeur connaît la clé secrète $\hat{\mathbf{s}}$, on obtient l'équation suivante :

$$h(\hat{\mathbf{z}} - \hat{\mathbf{s}}\mathbf{c}) = h(\hat{\mathbf{z}}' - \hat{\mathbf{s}}\mathbf{c}').$$

Pour conclure que $(\hat{\mathbf{z}} - \hat{\mathbf{s}}\mathbf{c}, \hat{\mathbf{z}}' - \hat{\mathbf{s}}\mathbf{c}')$ est une solution au problème Col$(h, D_\times)$, Lyubashevsky démontre les deux points suivants :

— avec les choix des paramètres $\hat{\mathbf{z}} - \hat{\mathbf{s}}\mathbf{c}$, $\hat{\mathbf{z}}' - \hat{\mathbf{s}}\mathbf{c}'$ sont dans $D_\times$.
— le schéma satisfait la propriété de témoin indistinguable et l'adversaire ne peut pas deviner exactement quelle clé a été utilisée pendant la phase d'apprentissage, par conséquent $\hat{\mathbf{z}} - \hat{\mathbf{s}}\mathbf{c} \neq \hat{\mathbf{z}}' - \hat{\mathbf{s}}\mathbf{c}'$ avec une probabilité au moins égale à $1/2$.

1.6.3 Le schéma d'identification de Kawachi-Tanaka-Xagawa

Dans cette section nous rappelons le schéma d'identification de Kawachi, Tanaka et Xagawa [KTX08], que nous nommons KTX, d'après ses auteurs. Ce protocole est basé sur un protocole de preuve de connaissance sans divulgation. Pour

chaque tour, la probabilité qu'un prouveur malhonnête parvienne à frauder est de $2/3$. Par conséquent, le protocole doit être exécuté un nombre de fois $\omega(\log n)$, afin d'obtenir une probabilité de triche exponentiellement petite en n.

Le prouveur $\mathcal{P}$ et le vérificateur $\mathcal{V}$ suivent un protocole où $\mathsf{COM}_{\boldsymbol{A}}$ est une fonction d'engagement (voir la section 1.4.2) définie par la matrice $\boldsymbol{A}$ obtenue grâce à l'algorithme $\mathsf{Id-init}$. Dans la suite du manuscrit, nous utilisons la notation COM au lieu de $\mathsf{COM}_{\boldsymbol{A}}$ et nous n'écrirons pas explicitement les vecteurs aléatoires utilisés pour calculer les engagements.

Description. Soient n le paramètre de sécurité et q, m deux entiers tels que $q = \mathsf{poly}(n)$ et $m \geq 10n \log q$. Le schéma proposé prend comme paramètre public partagé entre le prouveur et le vérificateur une matrice $\boldsymbol{A}$ choisie d'une manière uniformément aléatoire dans $\mathbb{Z}_q^{n \times m}$. La clé secrète du prouveur est un vecteur $\boldsymbol{x}$ choisi aléatoirement dans $\{0,1\}^m$ tel que $w_H(\boldsymbol{x}) = m/2$ et sa clé publique est le vecteur $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} \bmod q$. Le vérificateur vérifie que le prouveur connait un vecteur $\boldsymbol{x} \in \{0,1\}^m$ de poids de Hamming égal à $m/2$ et qui correspond à la clé publique $\boldsymbol{y}$ (c'est-à-dire tel que $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y} \bmod q$).

$\boldsymbol{A} \leftarrow \mathsf{Id-init}(1^n)$: Étant donné un paramètre de sécurité n. Choisir uniformément une matrice $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$.

$(\boldsymbol{y}, \boldsymbol{x}) \leftarrow \mathsf{Id-keygen}(\boldsymbol{A})$: Soit $\boldsymbol{A}$ une matrice obtenue grâce à l'algorithme $\mathsf{Id-init}$. L'algorithme génère une paire de clés de la manière suivante :
— Soit $\boldsymbol{x} \leftarrow \{0,1\}^m$, tel que $w_H(\boldsymbol{x}) = m/2$.
— Soit $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} \bmod q$.

$(\mathcal{P} \leftrightarrow \mathcal{V})$: Les paramètres en commun entre le prouveur et le vérificateur sont $(\boldsymbol{A}, \boldsymbol{y})$. Le prouveur possède la clé secrète $\boldsymbol{x}$ comme entrée auxiliaire.

1. $\mathcal{P}$ choisit $\pi \leftarrow S_m$ et $\boldsymbol{u} \leftarrow \mathbb{Z}_q^m$. $\mathcal{P}$ envoie c_0, c_1 et c_3 à $\mathcal{V}$ tels que
   — $c_1 = \mathsf{COM}(\pi \| \boldsymbol{A}\boldsymbol{u})$
   — $c_2 = \mathsf{COM}(\pi(\boldsymbol{u}))$
   — $c_3 = \mathsf{COM}(\pi(\boldsymbol{x} + \boldsymbol{u}))$
2. $\mathcal{V}$ choisit un challenge $ch \in \{1,2,3\}$ et il l'envoie à $\mathcal{P}$
3. — Si $ch = 1$, $\mathcal{P}$ envoie $\boldsymbol{e} = \pi(\boldsymbol{x})$ et $\boldsymbol{w} = \pi(\boldsymbol{u})$.
   — Si $ch = 2$, $\mathcal{P}$ envoie $\phi = \pi$ et $\boldsymbol{v} = \boldsymbol{x} + \boldsymbol{u}$.
   — Si $ch = 3$, $\mathcal{P}$ envoie $\varphi = \pi$ et $\boldsymbol{z} = \boldsymbol{u}$.

. Vérification :
— Si $ch = 1$, $\mathcal{V}$ vérifie si $c_2 = \mathsf{COM}(\boldsymbol{w})$, $c_3 = \mathsf{COM}(\boldsymbol{e} + \boldsymbol{w})$, $\boldsymbol{e} \in \{0,1\}^m$ et $w_H(\boldsymbol{e}) = m/2$.
— Si $ch = 2$, $\mathcal{V}$ vérifie si $c_1 = \mathsf{COM}(\phi \| \boldsymbol{A}\boldsymbol{v} - \boldsymbol{y})$ et $c_3 = \mathsf{COM}(\phi(\boldsymbol{v}))$.
— Si $ch = 3$, $\mathcal{V}$ vérifie si $c_1 = \mathsf{COM}(\varphi \| \boldsymbol{A}\boldsymbol{z})$ et $c_2 = \mathsf{COM}(\varphi(\boldsymbol{z}))$.
$\mathcal{V}$ retourne $\mathsf{desc} = 1$ si les tests sont valides, sinon il retourne $\mathsf{desc} = 0$.

Le lemme suivant énonce que pour toute paire de clés $(\boldsymbol{y}, \boldsymbol{x})$ provenant de l'algorithme de génération des clés $\mathsf{Id-keygen}$, il existe avec une probabilité écrasante, un vecteur $\boldsymbol{x}' \in \{0,1\}^m$ tel que $\boldsymbol{x}' \neq \boldsymbol{x}$, $\boldsymbol{A}\boldsymbol{x}' = \boldsymbol{A}\boldsymbol{x} = \boldsymbol{y} \bmod q$ et $w_H(\boldsymbol{x}') = m/2$.

Lemme 1.6. *(Lemme 4.3 [KTX08]) Pour tout $\boldsymbol{A}$ fixé, soit $Y := \{\boldsymbol{y} \in \mathbb{Z}_q^n |$ cardinal $(\{\boldsymbol{x} \in B(m, m/2) | \boldsymbol{A}\boldsymbol{x} = \boldsymbol{y} \bmod q\}) = 1\}$. Si $q^n/|B(m, m/2)|$ est négligeable en n, alors la probabilité d'obtenir $\boldsymbol{y} \in Y$, pour $(\boldsymbol{y}, \boldsymbol{x}) \leftarrow \mathsf{Id-keygen}(\boldsymbol{A})$ est négligeable en n.*

Sécurité. Avant de donner une analyse de sécurité, nous rappelons que KTX satisfait les deux propriétés suivantes :

1. Chaque clé publique $\boldsymbol{y}$ admet deux clés secrètes valides avec une probabilité écrasante (c'est-à-dire, il existe $\boldsymbol{x}, \boldsymbol{x}' \in \{0,1\}^m$, tels que $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}\boldsymbol{x}' = \boldsymbol{y} \bmod q$ et $w_H(\boldsymbol{x}) = w_H(\boldsymbol{x}') = m/2$). Ceci est vrai grâce au Lemme 1.6.
2. Le schéma est une preuve de connaissance à témoin indistinguable. En effet, d'après la Proposition 1.4, cette propriété est satisfaite puisque le schéma est une preuve de connaissance sans divulgation.

Maintenant, nous expliquons comment exploiter ces deux propriétés pour démontrer la sécurité de KTX contre les attaques parallèles (voir Définition 1.12). Au début du jeu, le challengeur obtient une matrice $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$ comme challenge, et son but est de résoudre le problème $\mathsf{SIS}_{n,q,m,\sqrt{m}}$ lié à $\boldsymbol{A}$ (c'est-à-dire trouver $\boldsymbol{s} \in \mathbb{Z}^m \setminus \{0\}$ tel que $\boldsymbol{A}\boldsymbol{s} = 0 \bmod q$ et $\|\boldsymbol{s}\|_2 \leq \sqrt{m}$) en utilisant l'attaquant.
Phase d'initialisation : Le challengeur génère une clé secrète $\boldsymbol{x} \in \{0,1\}^m$ telle que $w_H(\boldsymbol{x}) = m/2$, il calcule la clé publique $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} \bmod q$ et il donne $(\boldsymbol{y}, \boldsymbol{A})$ à l'attaquant.
Phase d'apprentissage : Pendant cette phase, le challengeur simule parfaitement le prouveur (honnête) puisqu'il connait le secret $\boldsymbol{x}$. Il pourra donc répondre correctement à tous les challenges envoyés par l'attaquant.
Phase d'imposture : En utilisant les réponses de l'attaquant (qui joue le rôle d'un prouveur malhonnête), le challengeur obtient une collision sur la fonction d'engagement COM, ou bien un vecteur $\boldsymbol{x}' \in \{0,1\}^m$ tel que $w_H(\boldsymbol{x}') = m/2$ et $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}' \bmod q$.

Par conséquent, le challengeur obtient $\boldsymbol{s} = \boldsymbol{x}' - \boldsymbol{x} \in \{-1, 0, 1\}^m$ qui est une solution au problème $\mathsf{SIS}_{n,q,m,\sqrt{m}}$ lié à la matrice challenge $\boldsymbol{A}$.

Pour finir il faut montrer que $\boldsymbol{s} \neq 0$, c'est-à-dire démontrer que $\boldsymbol{x}' \neq \boldsymbol{x}$. En effet, d'après le Lemme 1.6, nous savons qu'il existe deux secrets valides pour $\boldsymbol{y}$. De plus, puisque le schéma est à témoin indistinguable, l'attaquant ne peut rien savoir sur la clé secrète utilisée (dans notre cas c'est $\boldsymbol{x}$) pendant la phase d'apprentissage même lorsqu'il interagit en parallèle avec plusieurs instances du protocole. Par conséquent, nous avons que $\boldsymbol{x}' \neq \boldsymbol{x}$ avec une probabilité au moins égale à $1/2$.

Remarque 1.6. *En 2010, Cayrel, Lindner, Ruckert et Silva [CLRS10a] ont proposé une version en cinq passes du protocole* KTX*, que nous nommons* CLRS*, d'après ses*

auteurs. Comme démontré dans [CLRS10a], CLRS *est une preuve de connaissance sans divulgation. Pour chaque tour, la probabilité qu'un prouveur malhonnête parvienne à frauder est de* $1/2$*. Par conséquent, le protocole doit être exécuté un nombre de fois* $\omega(\log n)$*, afin d'obtenir une probabilité de triche négligeable. Une description formelle de ce schéma est donnée dans l'annexe A.*

1.7 Schémas de signature numériques basés sur les réseaux

Nous classons les schémas de signature basés sur les réseaux en deux familles. La première famille est celle des schémas avec trappe. Ces schémas sont obtenus suivant la méthode Hacher-et-Signer (pour *Hash-and-Sign*, en anglais) qui fut introduite par Diffie et Hellman dans [DH76]. Dans cette approche la clé publique est une fonction à trappe f et la clé de signature (c'est-à-dire la clé secrète) est f^{-1}. Pour signer un message μ, tout d'abord on calcule $y = H(\mu)$ un haché de μ où H est une fonction de hachage qui fournit des valeurs dans le domaine d'arrivée de f. Ensuite, on retourne $\sigma = f^{-1}(y)$ comme signature du message μ. La signature (μ, σ) est acceptée par l'algorithme de vérification si $f(\sigma) = H(\mu)$. La deuxième famille est celle des schémas sans trappe. Ses schémas sont obtenus grâce à la transformation de Fiat-Shamir [FS86] qui consiste à transformer un schéma d'identification en trois passes en un schéma de signature. L'idée de cette transformation est de remplacer l'interaction avec le vérificateur par un appel à un oracle aléatoire H. En effet, le signataire prend le rôle du prouveur dans un schéma d'identification et en utilisant H il remplace le challenge envoyé par le vérificateur par un haché des paramètres publics, de l'engagement et du message à signer.

1.7.1 Schémas de signatures avec trappe

Le premier schéma qu'on peut classer dans la famille des schémas de signatures avec trappe est celui de Gentry et al. [GPV08] présenté à STOC 2008. Dans ce schéma le signataire utilise l'algorithme TrapGen (voir la section 1.5.3.1) pour obtenir une paire de clés $(pk, sk) = (\boldsymbol{A}, \boldsymbol{S})$ où $\boldsymbol{S}$ est une base du réseau $\mathcal{L}_q^{\perp}(\boldsymbol{A})$, formée de vecteurs courts. La signature d'un message μ consiste en un vecteur $\boldsymbol{e} \in \mathbb{Z}^m$ généré en utilisant la matrice $\boldsymbol{S}$. La partie fondamentale de ce schéma est l'algorithme de génération des clés TrapGen. En effet, la taille du vecteur $\boldsymbol{e}$ dépend de la qualité de la base $\boldsymbol{S}$; plus les coefficients de $\boldsymbol{S}$ sont petits, plus la norme du vecteur $\boldsymbol{e}$ est petite. Dans [GPV08], les auteurs ont utilisé l'algorithme de génération des clés proposé par Ajtai [Ajt99]. Alwen et Peikert [AP11] et Micciancio et Peikert [MP12] ont proposé des améliorations de l'algorithme proposé par Ajtai

[Ajt99]. Ces travaux ont permis d'obtenir des tailles de signatures plus petites.

En remplaçant les réseaux q-aires par les réseaux idéaux, Stehlé et al. [SSTX09] ont donné une variante de la signature proposée dans [GPV08]. En utilisant la notion de *bonsai tree* [Pei09], Cash et al. [CHKP10] ont proposé un autre schéma de signature dans le modèle standard. La sécurité de ce schéma a été améliorée par Rückert [Rüc10]. En 2010, un autre schéma dans le modèle standard a été proposé par Boyen [Boy10].

Plus formellement, le schéma proposé par Gentry et al. [GPV08] se présente de la manière suivante :

Pramètres : Soient n le paramètre de sécurité, $m \geq (5+3\delta)n\log q$, $s \geq L \cdot \omega(\sqrt{\log m})$ (où $L = O(\sqrt{n \log q})$).

S$-$keygen(1^n) : Utiliser l'algorithme TrapGen(1^n) pour obtenir $(pk, sk) = (\boldsymbol{A}, \boldsymbol{S})$.

S$-$sign$(\boldsymbol{S}, \mu)$: Soit H un oracle aléatoire tel que $H : \{0,1\}^\star \to \mathbb{Z}_q^n$.

1. Soit $\boldsymbol{y} = H(\mu)$.
2. Soit $\boldsymbol{e} = \mathsf{SamplePre}(\boldsymbol{A}, \boldsymbol{S}, \boldsymbol{y}, s)$.
3. Retourner $\boldsymbol{e}$ comme signature du message μ.

S$-$verify$(\boldsymbol{A}, \boldsymbol{e}, \mu)$: Le vérificateur accepte la signature si $\|\boldsymbol{e}\|_2 \leq s\sqrt{m}$ et $\boldsymbol{A}\boldsymbol{e} = H(\mu) \bmod q$.

La sécurité de ce schéma est donnée par le théorème suivant.

Théorème 1.4 ([GPV08]). *S'il existe un attaquant $\mathcal{F}$ en temps polynomial qui peut casser la résistance à la contrefaçon existentielle (voir Définition 1.17) de ce schéma, alors il existe un algorithme en temps polynomial qui peut résoudre le problème* $\mathsf{SIS}_{n,q,m,2s\sqrt{m}}$ *dans le cas moyen.*

Démonstration. (idée) : Au début, le challengeur $\mathcal{C}$ reçoit une matrice $\boldsymbol{A} \leftarrow \mathbb{Z}_q^{n\times m}$ comme instance du problème SIS. Le challengeur initialise l'attaquant $\mathcal{F}$ avec $\boldsymbol{A}$ comme clé publique. On suppose que $\mathcal{F}$ envoie une requête de hachage à l'oracle H avant de demander la signature du message. Quand $\mathcal{F}$ envoie une requête $\mu \in \{0,1\}^\star$ à l'oracle H, le challengeur calcule $\boldsymbol{e}_\mu \leftarrow \mathsf{SampleDom}(1^n)$, enregistre $(\mu, \boldsymbol{e}_\mu)$ dans une liste L et retourne $\boldsymbol{A}\boldsymbol{e}_\mu \bmod q$ à $\mathcal{F}$ (Si H a déjà reçu la requête μ, le challengeur trouve $(\mu, \boldsymbol{e}_\mu)$ dans la liste L et retourne $\boldsymbol{A}\boldsymbol{e}_\mu \bmod q$). Lorsque $\mathcal{F}$ envoie une requête de signature pour un message $\mu \in \{0,1\}^\star$, le challengeur trouve $(\mu, \boldsymbol{e}_\mu)$ dans la liste et retourne $\boldsymbol{e}_\mu$ comme signature. À un certain moment, $\mathcal{F}$ retourne $(\mu^\star, \boldsymbol{e}^\star)$ comme signature contrefaite (On suppose que $\mathcal{F}$ a déjà demandé le haché du message $\mu^\star$ à l'oracle H avant de retourner sa réponse). Le challengeur $\mathcal{C}$ trouve $(\mu^\star, \boldsymbol{e}_{\mu^\star})$ dans la liste L et retourne $\boldsymbol{s} = \boldsymbol{e}^\star - \boldsymbol{e}_{\mu^\star}$ comme solution à l'instance SIS associée à la matrice $\boldsymbol{A}$ ($\boldsymbol{A}\boldsymbol{s} = 0 \bmod q$ car $\boldsymbol{A}\boldsymbol{e}^\star = \boldsymbol{A}\boldsymbol{e}_{\mu^\star} \bmod q$). Le vecteur $\boldsymbol{s}$ est de norme au plus égale à $2s\sqrt{m}$, car les deux vecteurs $\boldsymbol{e}^\star, \boldsymbol{e}_{\mu^\star}$ sont chacun de

norme au plus égale à $s\sqrt{m}$. En effet, $\boldsymbol{e}^\star$ est de norme au plus $s\sqrt{m}$, puisque la réponse de l'attaquant doit passer avec succès l'algorithme de vérification S−verify. De même, on a que $\boldsymbol{e}_{\mu^\star}$ est choisi à partir de la distribution gaussienne $\mathsf{D}_{\mathbb{Z}^m,s}$ (voir la section 1.5.3.1) par conséquent $\|\boldsymbol{e}_{\mu^\star}\|_2 \leq s\sqrt{m}$.

□

1.7.2 Schémas de signatures sans trappe

Nous allons voir à présent deux schémas de signatures proposés par Lyubashevsky dans [Lyu09, Lyu12] qui sont obtenus grâce à la transformation de Fiat-Shamir. Nous les nommons respectivement LSig09 et LSig12. Le schéma de signature dans [Lyu09], est obtenu par l'application de la transformation de Fiat-Shamir sur le schéma d'identification LID09 (voir la section 1.6.2). La sécurité du schéma sera développée en détail dans la section 1.7.2.1. Dans [Lyu12], Lyubashevsky a proposé un nouveau schéma de signature basé sur la difficulté du problème SIS et qui est une transformation de Fiat-Shamir d'une version améliorée du schéma d'identification LID08 (voir la section 1.6.1). Comme expliqué par Ducas-Binda dans sa thèse [DB13], la distribution uniforme sur $\{0, 1, \cdots, 5m-1\}^m$ de LID08 est remplacée par une distribution gaussienne et le vecteur $\boldsymbol{x}$ de clé secrète est remplacé par une matrice $\boldsymbol{S}$ avec des petits coefficients afin d'éviter les répétitions. D'autre part la taille de la matrice de la clé publique est considérablement réduite ; $m = 2n$ au lieu de $m = O(n \log q)$. Le schéma de signature LSig12 est décrit plus précisément dans la section 1.7.2.2.

1.7.2.1 Le schéma de signature LSig09

Le schéma de signature LSig09 a été proposé dans [Lyu09]. Dans sa thèse, Lyubashvesky [Lyu08b] a donné une analyse de sécurité complète pour ce schéma. Nous rappelons de manière formelle ce schéma qui est l'analogue du schéma de signature de Schnorr.

Soient k le paramètre de sécurité, $\mathcal{H}(\mathcal{D}, D_\times, m)$ une famille de fonctions de hachage comme dans la section 1.5.2.2 avec $D_\times = D_h$ et $m = 3\log n$.

S−keygen(1^k) :
- Soit $h \leftarrow \mathcal{H}(\mathcal{D}, D_\times, m)$.
- Soit $\hat{\mathbf{s}} \leftarrow D_{s,c}^m$.
- Soit $\mathbf{S} = h(\hat{\mathbf{s}})$.
- Retourner $(pk, sk) = ((\mathbf{S}, h), \hat{\mathbf{s}})$

S−sign$(h, \mathbf{S}, \hat{\mathbf{s}}, \mu)$: Soient un message $\mu \in \{0,1\}^\star$ et H_L un oracle aléatoire tel que $H_L : \{0,1\}^\star \to D_{s,c}$

1. Soit $\hat{\mathbf{y}} \leftarrow D_y^m$.
2. Soit $\mathbf{e} = H_L(h(\hat{\mathbf{y}}), \mu)$.

	Définition
n	puissance de 2 plus grande que le paramètre de sécurité k
p	premier de l'ordre de $\Theta(n^4)$
m	$3 \log n$
$\mathcal{D}$	l'anneau $\mathbb{Z}_p[x]/\langle x^n + 1\rangle$
D_h	$\{\mathbf{g} \in \mathcal{D} : \|\mathbf{g}\|_\infty \leq mn^{1.5} \log n + \sqrt{n} \log n\}$
D_y	$\{\mathbf{g} \in \mathcal{D} : \|\mathbf{g}\|_\infty \leq mn^{1.5} \log n\}$
D_z	$\{\mathbf{g} \in \mathcal{D} : \|\mathbf{g}\|_\infty \leq mn^{1.5} \log n - \sqrt{n} \log n\}$
$D_{s,c}$	$\{\mathbf{g} \in \mathcal{D} : \|\mathbf{g}\|_\infty \leq 1\}$

TABLE 1.2 – Ensembles et paramètres (Figure 6.1 dans [Lyu08b]).

3. Soit $\hat{\mathbf{z}} = \hat{\mathbf{s}}\mathbf{e} + \hat{\mathbf{y}}$.
4. Si $\hat{\mathbf{z}} \in D_z^m$, alors retourner $\sigma = (\hat{\mathbf{z}}, \mathbf{e})$. Sinon, aller à l'étape 1.

S$-$verify$(h, \mathbf{S}, \sigma = (\hat{\mathbf{z}}, \mathbf{e}), \mu)$: Si $\hat{\mathbf{z}} \in D_z^m$ et $\mathbf{e} = H_L(h(\hat{\mathbf{z}}) - \mathbf{S}\mathbf{e}, \mu)$, alors retourner 1, sinon retourner 0.

La sécurité repose sur les deux résultats suivants :

Lemme 1.7 (Lemme 5-2 dans [Lyu08b]). *Soit $h \in \mathcal{H}(\mathcal{D}, D_\times, m)$. Pour tout secret $\hat{\mathbf{s}}_0$ choisi d'une manière uniformément aléatoire dans $D_{s,c}^m$, la probabilité qu'il n'existe pas $\hat{\mathbf{s}}_1 \in D_{s,c}^m$ tels que, $\hat{\mathbf{s}}_0 \neq \hat{\mathbf{s}}_1$ et $h(\hat{\mathbf{s}}_0) = h(\hat{\mathbf{s}}_1)$ est négligeable en le paramètre de sécurité.*

Démonstration. D'après les paramètres de la Table 1.2, nous avons que le cardinal de l'ensemble d'arrivée de h (c'est-à-dire $\mathcal{D}$) est p^n et le cardinal de l'ensemble $D_{s,c}^m$ est 3^{mn}. Ceci implique qu'au plus p^n éléments de $D_{s,c}^m$ ne produisent pas de collision pour h. Par conséquent, pour $\hat{\mathbf{s}}_0$ choisi au hasard dans $D_{s,c}^m$, la probabilité qu'il n'existe pas $\hat{\mathbf{s}}_1 \in D_{s,c}^m$ tel que, $\hat{\mathbf{s}}_0 \neq \hat{\mathbf{s}}_1$ et $h(\hat{\mathbf{s}}_0) = h(\hat{\mathbf{s}}_1)$ est

$$\left(\frac{p}{3^m}\right)^n = 2^{-\Omega(n \log n)}.$$

□

Théorème 1.5 (Théorème 6-5 dans [Lyu08b]). *Pour tous $h \in \mathcal{H}(\mathcal{D}, D_\times, m)$, $\mu \in \{0,1\}^\star$ et $\hat{\mathbf{s}}, \hat{\mathbf{s}}' \in D_{s,c}^m$ tels que $h(\hat{\mathbf{s}}) = h(\hat{\mathbf{s}}') = \mathbf{S}$, nous avons que*

$$\Delta((\hat{\mathbf{z}}, \mathbf{e}), (\hat{\mathbf{z}}', \mathbf{e}')) = n^{-\omega(1)}.$$

Où $(\hat{\mathbf{z}}, \mathbf{e}) \leftarrow$ S$-$sign$(h, \mathbf{S}, \hat{\mathbf{s}}, \mu)$ et $(\hat{\mathbf{z}}', \mathbf{e}') \leftarrow$ S$-$sign$(h, \mathbf{S}, \hat{\mathbf{s}}', \mu)$.

Remarque 1.7. *La preuve de résistance à la contrefaçon de la signature* LSig09 *est divisée en deux parties. Tout d'abord Lyubashvesky montre que le problème de trouver une collision sur une fonction de hachage de la famille $\mathcal{H}(\mathcal{D}, D_\times, m)$ se réduit au problème de contrefaire une signature du schéma* LSig09*. Autrement dit, il montre que s'il*

existe un attaquant qui peut contrefaire des signatures, alors ce dernier peut être utilisé pour trouver une collision sur une fonction de hachage de la famille $\mathcal{H}(\mathcal{D}, D_\times, m)$. Ensuite, pour montrer que ceci repose sur un problème difficile des réseaux, il utilise le résultat (Théorème 1.3) qui réduit le problème de résoudre (x^n+1)-SVP$_\gamma^\infty$ (pour un certain γ) dans le pire cas, au problème de trouver des collisions sur une fonction de hachage choisie aléatoirement dans $\mathcal{H}(\mathcal{D}, D_\times, m)$.

Le théorème suivant résume la première partie de la preuve de résistance à la contrefaçon.

Théorème 1.6 (Théorème 6.6 dans [Lyu08b]). *S'il existe un algorithme en temps polynomial $\mathcal{F}$ qui peut casser la résistance à la contrefaçon existentielle de ce schéma (voir Définition 1.17). Alors pour une fonction h choisie d'une manière uniformément aléatoire dans $\mathcal{H}(\mathcal{D}, D_\times, m)$, il existe un algorithme en temps polynomial $\mathcal{A}$ qui trouve une solution pour le problème* Col$(h, D_\times)$.

Démonstration. La preuve de ce théorème est donnée dans l'annexe B. □

La sécurité du schéma est basée sur la difficulté du problème (x^n+1)-SVP$_\gamma^\infty$. En effet, en combinant le théorème précédant avec le Théorème 1.3 nous obtenons le corollaire suivant.

Corollaire 1.1 (Corollaire 6.7 dans [Lyu08b]). *S'il existe un algorithme en temps polynomial $\mathcal{F}$ qui peut casser la résistance à la contrefaçon existentielle de ce schéma, alors il existe un algorithme en temps polynomial qui permet de résoudre (x^n+1)-SVP$_\gamma^\infty$ dans le pire cas pour $\gamma = \tilde{O}(n^{2.5})$.*

1.7.2.2 Le schéma de signature LSig12

Définition 1.34 ([Lyu12]). *Soit m un entier positif. La distribution continue de la loi normale centrée en $\boldsymbol{v}$ d'écart-type σ sur $\mathbb{R}^m$ est définie par la fonction*

$$\rho_{\boldsymbol{v},\sigma}^m(\boldsymbol{x}) = \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^m e^{\frac{-\|\boldsymbol{x}-\boldsymbol{v}\|_2^2}{2\sigma^2}}.$$

Lorsque $\boldsymbol{v} = 0$, on utilise la notation ρ_σ^m.
La distribution discrète de la loi normale centrée en $\boldsymbol{v} \in \mathbb{Z}^m$ d'écart-type σ sur $\mathbb{Z}^m$ est définie par $\mathsf{D}_{\boldsymbol{v},\sigma}^m(\boldsymbol{x}) = \frac{\rho_{\boldsymbol{v},\sigma}^m(\boldsymbol{x})}{\rho_\sigma^m(\mathbb{Z}^m)}$ *avec $\rho_\sigma^m(\mathbb{Z}^m) = \sum_{\boldsymbol{z}\in\mathbb{Z}^m} \rho_\sigma^m(\boldsymbol{z})$. Lorsque $\boldsymbol{v} = 0$, on utilise la notation* D_σ^m.

Soit n le paramètre de sécurité. Le schéma de signature LSig12 se présente de la manière suivante :

S$-$keygen(1^n) :
— Soit $\boldsymbol{S} \leftarrow \{-d, \cdots, 0, \cdots, d\}^{m\times k}$.

— Soit $\boldsymbol{A} \leftarrow \mathbb{Z}_q^{n \times m}$.
— Soit $\boldsymbol{T} = \boldsymbol{AS}$.
— Retourner $(pk, sk) = ((\boldsymbol{A}, \boldsymbol{T}), \boldsymbol{S})$.

S−sign$(\boldsymbol{A}, \boldsymbol{T}, \boldsymbol{S}, \mu)$: Soient un message $\mu \in \{0,1\}^\star$ et H un oracle aléatoire tel que $H : \{0,1\}^\star \to \{\boldsymbol{v} : \boldsymbol{v} \in \{-1,0,1\}^k, \|\boldsymbol{v}\|_1 \leq \kappa\}$.

1. Soit $\boldsymbol{y} \leftarrow \mathsf{D}_\sigma^m$.
2. Soit $\boldsymbol{e} = H(\boldsymbol{Ay}, \mu)$.
3. Soit $\boldsymbol{z} = \boldsymbol{Se} + \boldsymbol{y}$.
4. Retourner $(\boldsymbol{z}, \boldsymbol{e})$ avec une probabilité $\min(1, \frac{\mathsf{D}_\sigma^m(\boldsymbol{z})}{M\mathsf{D}_{\boldsymbol{Se},\sigma}^m(\boldsymbol{z})})$.

S−verify$(\boldsymbol{A}, \boldsymbol{T}, (\boldsymbol{z}, \boldsymbol{e}), \mu)$: Si $\|\boldsymbol{z}\|_2 \leq \eta\sigma\sqrt{m}$ et $\boldsymbol{e} = H(\boldsymbol{Az} - \boldsymbol{Te}, \mu)$, alors retourner 1, sinon retourner 0.

	Définition	Valeur
n	puissance de 2	512
q	un entier	2^{18}
d	-	1
k	-	512
η	-	1.3
m	$2n$	1024
κ	$2^\kappa \cdot \binom{\kappa}{n} \geq 2^{100}$	14
σ	$\approx 6 \cdot d \cdot \kappa \cdot \sqrt{m}$	2688
M	$\approx= exp(12d\kappa\sqrt{m}/\sigma + (d\kappa\sqrt{m}/2\sigma)^2)$	7.4

TABLE 1.3 – Les paramètres dans la dernière colonne correspondent à ceux donnés dans la colonne V de la Figure 2 dans [Lyu12].

Théorème 1.7 ([Lyu12]). *S'il existe un algorithme en temps polynomial $\mathcal{F}$ qui peut casser la résistance à la contrefaçon existentielle (voir Définition 1.17) de ce schéma, alors il existe un algorithme en temps polynomial qui peut résoudre le problème* $\mathsf{SIS}_{n,q,m,\beta}$ *dans le cas moyen pour* $\beta = \tilde{O}(n)$.

Nous renvoyons le lecteur à [Lyu12] pour le détail de la preuve de sécurité.

Chapitre 2

Un schéma de signature de cercle

Sommaire

Chapitre 2

Un schéma de signature de cercle

La notion de signature de cercle a été introduite en 2001 par Rivest, Shamir, et Tauman [RST01], elle permet à un utilisateur de signer anonymement des messages au non d'une entité à laquelle il appartient. Dans un tel schéma, chaque utilisateur a une paire de clés ; une clé secrète pour signer le message et une clé publique pour vérifier la signature. Un utilisateur peut choisir un ensemble de clés, que nous appelons un cercle, et signe au non des utilisateurs associés, sans demander de permission ou d'autorisation. Cette signature peut être vérifiée en utilisant les clés publiques des utilisateurs qui forment le cercle associé à la signature. Un schéma de signature de cercle doit satisfaire la notion classique d'inforgeabilité ainsi que la notion d'anonymat. La première propriété assure qu'il doit être impossible de signer au non d'un ensemble sans connaitre au moins une des clés secrètes associées. La deuxième propriété garantit que l'on peut pas connaitre l'identité réelle du signataire en regardant les signatures.

En transformant le schéma de signature classique de Lyubashevsky LSig09 (voir la section 1.7.2.1), nous avons proposé à la conférence Africacrypt'13, dans un travail commun avec Carlos Aguilar Melchor, Xavier Boyen, Laurent Fousse et Philippe Gaborit [AMBB+13], un nouveau schéma de signature de cercle basé sur les réseaux. Notre schéma satisfait une propriété qui est plus forte que la notion classique d'anonymat contre la révélation totale des clés. En effet, nous montrons que même si la paire de clés du signataire est choisie par un attaquant (en ayant par exemple remplacé celle-ci grâce à un accès temporaire en écriture à un dossier contenant cette paire), la distance statistique entre les signatures générées par cet utilisateur et celles générées par les autres reste négligeable.

En ce qui concerne l'inforgeabilité, le meilleur schéma de signature de cercle basé sur les réseaux satisfait l'inforgeabilité contre les attaques à sous cercles choisis ou bien les attaques avec corruption interne avec des cercles de taille logarithmique en la valeur du paramètre de sécurité. Nous présentons deux variantes de notre schéma. Le schéma de base satisfait l'inforgeabilité contre les attaques à

sous cercles choisis. En augmentant la taille de signature et des clés par un facteur k (k étant le paramètre de sécurité, typiquement 80-100), nous obtenons une variante qui satisfait l'inforgeabilité contre les attaques avec corruption interne pour des cercles de taille arbitraire. La technique utilisée est assez générale et peut être adaptée à d'autres schémas existants.

Organisation du chapitre. Ce chapitre est organisé de la manière suivante. Dans la section 2.1, nous donnons des définitions formelles du modèle de sécurité pour les schémas de signature de cercle et nous introduisons une nouvelle notion d'anonymat. Dans la section 2.2, nous rappelons deux schémas de signature de cercle basés sur les réseaux. Dans la section 2.3, nous présentons notre schéma de signature de cercle et nous analysons sa sécurité dans la section 2.4. Dans la section 2.5, nous proposons une nouvelle technique pour obtenir l'inforgeabilité contre corruption interne.

Notations supplémentaires. Chaque paire de clés du schéma de signature de cercle est définie uniquement par un entier qui est son indice. Nous définissons un cercle R comme étant un ensemble des clés publiques. Nous considérons qu'il existe une bijection entre les utilisateurs et les paires de clés, nous utilisons cette bijection implicitement pour dire qu'un utilisateur appartient à un cercle. Nous notons $\#R$ la taille du cercle (c'est-à-dire le nombre de clés de vérification qu'il contient) et par $index(R)$ l'ensemble des entiers correspondant aux indices des clés publiques dans R.

Nous décrivons un cercle $R = \{pk_{i_1}, \cdots, pk_{i_{\#R}}\}$, par exemple lorsqu'il est une entrée pour un oracle aléatoire, par $desc(pk_{i_1}) \| \cdots \| desc(pk_{i_{\#R}})$ avec $desc(pk)$ est la description binaire de la clé publique et $i_1 < \cdots < i_{\#R}$ (par conséquent la représentation est unique). Lorsque c'est possible, nous ignorons $desc()$ pour ces notations.

2.1 Définitions et modèle de sécurité

En 2006, Bender, Katz, et Morselli [BKM06] ont revisité les notions de sécurité des schémas de signature de cercle et ils ont donné des nouvelles définitions permettant d'avoir une sécurité contre des attaques différentes. Pour l'inforgeabilité, le nouveau modèle de sécurité proposé garantit la sécurité dans des scénarios où l'attaquant fabrique lui même des clés publiques (il construit ses propres cercles) ou bien dans les cas où il obtient quelques clés secrètes valides. D'autre part, Bender et al., ont introduit la notion d'anonymat contre la révélation totale des clés. C'est le cas où toutes les clés secrètes des utilisateurs sont rendues public. Bien sûr dans ce cas, il n'est plus possible d'avoir la propriété d'inforgeabilité, par contre l'anonymat des signatures générées ultérieurement est préservé.

Une telle forte propriété d'anonymat est une garantie rassurante pour les utilisateurs qui hésitent à divulguer un secret, particulièrement si les conséquences d'une identification sont désastreuses. Il semble raisonnable d'avoir une notion d'anonymat aussi forte, en particulier si l'anonymat doit être préservé pour des décennies. Par conséquent, l'anonymat n'est plus basé sur l'estimation de la complexité de calcul et on parle d'anonymat inconditionnel.

2.1.1 Définitions

2.1.1.1 Variantes de la signature de cercle

Dans cette section, nous rappelons deux variantes de la signature de cercle. Il s'agit de deux schémas qui permettent de générer anonymement des signatures dans un contexte de multi-utilisateurs.

Signature de groupe

Les signatures de groupe ont été introduites en 1991 par Chaum et van Heyst [CVH91], elles permettent à un utilisateur de signer anonymement au non d'un groupe de personnes. Dans les schémas de signature de groupe il existe un mécanisme de révocation d'anonymat qui permet à une entité appelée juge de révéler l'identité du signataire en cas de conflit. Contrairement aux schémas de signature de cercle, les schémas de signature de groupe admettent un manager qui gère les utilisateurs. Par conséquent pour signer au nom d'un groupe, l'utilisateur doit tout d'abord obtenir une permission de la part de son manager.

Signature de cercle à seuil

La notion de signature de cercle à seuil a été introduite par Bresson, Stern et Szydlo [BSS02] à CRYPTO' 2002. Pour un cercle de taille N (c'est-à-dire de N utilisateurs), un schéma de signature de cercle à seuil permet à t utilisateurs parmi N de générer une signature au nom de tous les membres du cercle. La vérification de la signature donne une preuve incontestable qu'exactement t utilisateurs ont participé à la génération de la signature. Les signatures de cercle à seuil sont une généralisation des signatures de cercle.

2.1.1.2 Signature de cercle

Nous donnons une définition formelle de ce que c'est un schéma de signature de cercle.

Définition 2.1. *[BKM06] Un schéma de signature de cercle consiste en un triplet d'algorithmes (*Ring−gen*,* Ring−sign*,* Ring−verify*) :*

Ring−gen(1^k) : *est un algorithme probabiliste polynomial qui prend un paramètre de sécurité 1^k comme entrée et retourne une paire de clés (pk, sk), où pk est une clé publique et sk est une clé secrète. Pour plusieurs schémas, l'utilisateur doit partager des paramètres publics dérivés de 1^k avec les autres utilisateurs du même cercle. Par conséquent, nous supposons que* Ring−gen(1^k) *admet deux algorithmes auxiliaires :* Ring−gen−params(1^k) *il génère un ensemble de paramètres publics que nous notons par $\mathcal{P}$, cet ensemble est utilisé par tous les algorithmes ; et* Ring−gen−keys$(\mathcal{P})$ *qui génère une paire de clés en utilisant l'ensemble des paramètres publics $\mathcal{P}$.*

Ring−sign$(\mathcal{P}, sk_{i_0}, \mu, R)$: *est un algorithme probabiliste polynomial qui prend en entrée un ensemble $\mathcal{P}$, un cercle $R = \{pk_i\}_{i\in[\ell]}$ (ℓ est polynomial en k), une clé de signature sk_{i_0} telle qu'il existe une clé publique $pk_{i_0} \in R$ qui correspond à sk_{i_0}, et un message $\mu \in \{0,1\}^\star$. Il retourne une signature σ du message μ suivant la clé secrète sk_{i_0}, ou bien* failed.

Ring−verify$(\mathcal{P}, \sigma, \mu, R)$: *est un algorithme déterministe qui prend en entrée l'ensemble $\mathcal{P}$, une signature de cercle σ et un cercle R, il retourne 1 si σ est une signature de cercle valide du message μ et 0 sinon.*

Un schéma de signature de cercle doit satisfaire la propriété de complétude suivante.

Complétude. Pour tous k, ℓ (ℓ est polynomial en k), $\mathcal{P} \in$ Ring−gen−params(1^k), $\{(pk_i, sk_i)\}_{i\in[\ell]} \subset$ Ring−gen−keys$(\mathcal{P})$, $i_0 \in [\ell]$, $\mu \in \{0,1\}^\star$, et $\sigma \in$ Ring−sign$(\mathcal{P}, sk_{i_0}, \mu, \{pk_i\}_{i\in[\ell]})$, telle que $\sigma \neq$ failed on a, Ring−verify$(\mathcal{P}, \sigma, \mu, \{pk_i\}_{i\in[\ell]}) = 1$.

2.1.2 Modèle de sécurité

Comme nous l'avons indiqué, la sécurité de ces signatures repose sur deux propriétés fondamentales : l'inforgeabilité et l'anonymat. Nous utilisons les définitions de Bender et al. [BKM06] dans le but de donner un modèle de sécurité formel pour les schémas de signature de cercle.

2.1.2.1 Inforgeabilité

Dans [BKM06], Bender et al., ont introduit trois notions différentes d'inforgeabilité. Nous allons les rappeler en utilisant un ordre croissant en terme de sécurité. Nous décrivons ces définitions en utilisant un jeu entre un challengeur $\mathcal{C}$ et un attaquant $\mathcal{F}$.

La première définition est la plus faible. Il s'agit du cas où les requêtes pour l'oracle de signature et la signature forgée de l'attaquant sont faites suivant un cercle fixé au début du jeu.

Définition 2.2. *(Inforgeabilité contre les attaques à cercle fixe) Soient k le paramètre de sécurité et ℓ un entier polynomial en k. Le challengeur $\mathcal{C}$ est donné $\mathcal{P} \leftarrow \mathsf{Ring-gen-params}(1^k)$.*

- ***Phase d'initialisation :*** *Le challengeur $\mathcal{C}$ exécute ℓ fois l'algorithme $\mathsf{Ring-gen-keys}(\mathcal{P})$ et obtient $\{(pk_i, sk_i)\}_{i\in[\ell]}$. Ensuite il envoie l'ensemble $R = \{pk_i\}_{i\in[\ell]}$ à l'attaquant $\mathcal{F}$.*
- ***Phase de requêtes :*** *$\mathcal{F}$ peut faire un nombre polynomial de requêtes de signature. Les requêtes sont de la forme (i_{signer}, μ, R) avec $\mu \in \{0,1\}^\star$, $i_{signer} \in index(R)$. Le challengeur répond par $\sigma \leftarrow \mathsf{Ring-sign}(\mathcal{P}, sk_{i_{signer}}, \mu, R)$.*
- ***Phase de réponse :*** *$\mathcal{F}$ retourne $(\sigma^\star, \mu^\star, R)$. L'attaquant gagne le jeu si $\mathsf{Ring-verify}(\mathcal{P}, \sigma^\star, \mu^\star, R) = 1$ et si le message $\mu^\star$ n'apparait pas dans les requêtes envoyées pendant la phase de requêtes.*

Un schéma de signature de cercle satisfait la propriété d'inforgeabilité contre les attaques à cercle fixe si la probabilité qu'un attaquant puisse gagner ce jeu est négligeable en le paramètre de sécurité k.

La définition suivante est plus forte que la définition précédente, c'est l'inforgeabilité contre les attaques à sous-cercles choisis. Dans ce cas, les requêtes pour l'oracle de signature et la signature forgée de l'attaquant sont faites suivant n'importe quel cercle $S \subset R$.

Définition 2.3. *(Inforgeabilité contre les attaques à sous-cercles choisis) Soient k le paramètre de sécurité et ℓ un entier polynomial en k. Le challengeur $\mathcal{C}$ est donné $\mathcal{P} \leftarrow \mathsf{Ring-gen-params}(1^k)$.*

- ***Phase d'initialisation :*** *Le challengeur $\mathcal{C}$ exécute ℓ fois l'algorithme $\mathsf{Ring-gen-keys}(\mathcal{P})$ et obtient $\{(pk_i, sk_i)\}_{i\in[\ell]}$. Ensuite il envoie l'ensemble $R = \{pk_i\}_{i\in[\ell]}$ à l'attaquant $\mathcal{F}$.*
- ***Phase de requêtes :*** *$\mathcal{F}$ peut faire un nombre polynomial de requêtes de signature. Les requêtes sont de la forme (i_{signer}, μ, S) avec $\mu \in \{0,1\}^\star$, $i_{signer} \in index(S)$ et $S \subset R$. Le challengeur répond par $\sigma \leftarrow \mathsf{Ring-sign}(\mathcal{P}, sk_{i_{signer}}, \mu, S)$.*
- ***Phase de challenge :*** *$\mathcal{F}$ retourne $(S^\star, \mu^\star, \sigma^\star)$ et gagne le jeu si $S^\star \subseteq R$, $\mathsf{Ring-verify}(\sigma^\star, \mu^\star, S^\star) = 1$ et si aucune requête de la forme $(*, \mu^\star, S^\star)$ [1] n'a pas été effectuée pendant la phase de requêtes.*

Un schéma de signature de cercle satisfait la propriété d'inforgeabilité contre les attaques à sous-cercles choisis si la probabilité qu'un attaquant puisse gagner ce jeu est négligeable en le paramètre de sécurité k.

1. Une requête de la forme $(*, \mu^\star, S^\star)$ est une requête dans l'ensemble $\{(i_{signer}, \mu^\star, S^\star) : i_{signer} \in index(S^\star)\}$

La définition suivante est la plus forte, c'est inforgeabilité contre corruption interne. Il s'agit d'une définition qui prend en compte plusieurs scénarios d'attaques. Dans cette configuration, l'attaquant peut demander des requêtes de signature suivant n'importe quel cercle (c'est-à-dire que l'attaquant peut ajouter à ces cercles des clés publiques générées par lui même). L'attaquant est également donné accès à un oracle de corruption qui pour tout $i \in index(R)$ retourne sk_i. Plus formellement, nous avons la définition suivante :

Définition 2.4. *(Inforgeabilité contre corruption interne) Soient k le paramètre de sécurité, ℓ un entier polynomial en k et C l'ensemble d'utilisateurs corrompus $C \leftarrow \emptyset$. Le challengeur $\mathcal{C}$ est donné $\mathcal{P} \leftarrow \mathsf{Ring-gen-params}(1^k)$.*

- ***Phase d'initialisation :*** *Le challengeur $\mathcal{C}$ exécute ℓ fois l'algorithme $\mathsf{Ring-gen-keys}(\mathcal{P})$ et obtient $\{(pk_i, sk_i)\}_{i\in[\ell]}$. Ensuite il envoie l'ensemble $R = \{pk_i\}_{i\in[\ell]}$ à l'attaquant $\mathcal{F}$.*
- ***Phase de requêtes :*** *$\mathcal{F}$ peut faire deux types de requêtes :*
  - *Des requêtes de signature sont de la forme (i_{signer}, μ, S) avec $i_{signer} \in index(S) \cap index(R)$. La réponse est la signature $\sigma \leftarrow \mathsf{Ring-sign}(\mathcal{P}, sk_{i_{signer}}, \mu, S)$.*
  - *Des requêtes de corruption qui sont de la forme (i) telle que $i \in index(R)$, le challengeur $\mathcal{C}$ retourne la clé secrète sk_i et rajoute pk_i à l'ensemble C.*
- ***Phase de réponse :*** *$\mathcal{F}$ retourne $(S^\star, \mu^\star, \sigma^\star)$ et gagne le jeu si $\mathsf{Ring-verify}(\mathcal{P}, \sigma^\star, \mu^\star, S^\star) = 1$, aucune requête de la forme $(*, \mu^\star, S^\star)$ ne figure parmi les requêtes de signature et $S^\star \subseteq R \setminus C$.*

Un schéma de signature de cercle satisfait la propriété d'inforgeabilité contre corruption interne si la probabilité qu'un attaquant puisse gagner ce jeu est négligeable en le paramètre de sécurité k.

2.1.2.2 Anonymat

Dans [BKM06], Bender et al. ont défini différents niveaux de sécurité pour l'anonymat des schémas de signature de cercle. La notion, la plus forte, est l'anonymat contre l'exposition totale des clés. Plus précisément, ils ont utilisé la même définition pour formaliser deux niveaux de sécurité à la fois (il s'agit de l'anonymat contre les attaques d'attribution et l'anonymat contre la révélation totale des clés) ce qui a donnée une définition un peu trop complexe.

Nous rappelons formellement ces deux niveaux de sécurité. Nous considérons le jeu suivant entre un challengeur $\mathcal{C}$ et un attaquant $\mathcal{A}$.

Définition 2.5. *Soient k le paramètre de sécurité et ℓ un entier polynomial en k. Le challengeur $\mathcal{C}$ est donné $\mathcal{P} \leftarrow \mathsf{Ring-gen-params}(1^k)$.*

- ***Phase d'initialisation :*** *Le challengeur $\mathcal{C}$ exécute ℓ fois l'algorithme $\mathsf{Ring-gen-keys}(\mathcal{P})$ et obtient $\{(pk_i, sk_i)\}_{i\in[\ell]}$. Ensuite il envoie l'ensemble $R = \{pk_i\}_{i\in[\ell]}$ à l'attaquant $\mathcal{F}$.*

— ***Phase de requêtes :*** *$\mathcal{A}$ peut faire des requêtes de signature. Une requête est de la forme (i_{signer}, μ, S) avec $i_{signer} \in index(S) \cap index(R)$ (S n'est pas forcément un sous ensemble de R). La réponse est la signature $\sigma \leftarrow \mathsf{Ring{-}sign}(\mathcal{P}, sk_{i_{signer}}, \mu, S)$.*

— ***Phase de challenge :*** *$\mathcal{A}$ renvoie $(\mu^\star, S^\star, i_0, i_1)$ comme challenge, avec $\mu^\star \in \{0,1\}^\star$ et $S^\star$ est un cercle tel que $(pk_{i_0}, pk_{i_1}) \in S^\star \cap R$. Le cercle $S^\star$ n'est pas forcément inclus dans le cercle R. Le challengeur envoie $\{sk_i\}_{i \neq i_0}$ à $\mathcal{A}$. De plus, le challengeur choisit un bit $b \in \{0,1\}$ et il envoie la signature $\sigma \leftarrow \mathsf{Ring{-}sign}(\mathcal{P}, sk_{i_b}, \mu^\star, S^\star)$ à l'attaquant.*

— ***Phase de réponse :*** *L'attaquant retourne un bit b' et gagne le jeu si $b' = b$.*

Un schéma de signature de cercle satisfait la propriété d'anonymat contre les attaques d'attribution si la probabilité qu'un attaquant puisse gagner ce jeu est négligeablement proche de $1/2$. Si dans la phase de challenge, l'attaquant est donné $\{sk_i\}_{i \in [\ell]}$, alors le schéma satisfait la propriété d'anonymat contre la révélation totale des clés.

Maintenant nous introduisons une nouvelle définition pour l'anonymat qui est plus forte et moins complexe. En effet, nous n'avons pas besoin de définir un oracle de signature ou bien de corruption pour l'attaquant. Ceci est dû au fait que l'attaquant, dans notre modèle, connait toutes les clés secrètes et donc il peut lui même simuler les deux oracles. Également, nous n'avons pas besoin de la première étape dans laquelle le challengeur doit générer les paramètres et les clés, puisque ces derniers peuvent être choisis par l'attaquant. Par conséquent, nous parlons d'anonymat inconditionnel contre les attaques à paramètres choisis.

Nous considérons le jeu suivant entre un challenger $\mathcal{C}$ et un adversaire $\mathcal{A}$

Définition 2.6. *(Anonymat inconditionnel contre les attaques à paramètres choisis)*

1. *L'adversaire $\mathcal{A}$ envoie au challengeur $\mathcal{C}$:*
   — *l'ensemble des paramètres publics $\mathcal{P}$,*
   — *un cercle $R = \{pk_i\}_{i \in [\ell]}$ pour ℓ polynomial en k,*
   — *deux indices distincts $i_0, i_1 \in [\ell]$,*
   — *deux clés secrètes sk_{i_0}, sk_{i_1}, et un message $\mu \in \{0,1\}^\star$.*
2. *$\mathcal{C}$ génère deux signatures $\sigma_0 \leftarrow \mathsf{Ring{-}sign}(\mathcal{P}, sk_{i_0}, \mu, R)$, $\sigma_1 \leftarrow \mathsf{Ring{-}sign}(\mathcal{P}, sk_{i_1}, \mu, R)$ et choisit un bit aléatoire b. Si $\sigma_0 \neq$ `failed` et $\sigma_1 \neq$ `failed`, il envoie σ_b à $\mathcal{A}$, sinon le jeu recommence.*
3. *L'adversaire $\mathcal{A}$ retourne un bit b' et gagne le jeu si $b' = b$.*

Nous définissons l'avantage de l'adversaire $\mathcal{A}$ comme étant

$$\mathsf{Adv}_{\mathcal{A}} = |\mathsf{Pr}\,[b' = b] - \frac{1}{2}|.$$

Un schéma de signature de cercle vérifie l'anonymat inconditionnel contre les attaques à paramètres choisis si tout adversaire $\mathcal{A}$ a un avantage $\mathsf{Adv}_{\mathcal{A}}$ négligeable en le paramètre de sécurité k.

2.2 Schémas de signature de cercle basés sur les réseaux

La plupart des schémas de signature de cercle qui existent dans la littérature sont basés sur la théorie de nombre : le problème de factorisation des grand nombres entiers [RST01, DKNS04], le problème de logarithme discret [AOS02, HS03] et des problèmes sur le pairing [SW07, Boy07].

Dans cette section nous nous intéressons à deux schémas de signature de cercle basés sur les réseaux. En 2010, Brakersi et Kalai [BK10] ont proposé un schéma de signature de cercle en utilisant des fonctions à trappe particulièrement utiles pour la construction d'outils cryptographiques collaboratifs, qu'ils sont baptisé fonctions à trappe de cercle (voir la section 1.5.3.3 pour plus de détails). Leur schéma a une sécurité basée sur le problème SIS dans le modèle standard. En 2011, Wang et Sun [WS11] ont proposé deux schémas de signature de cercle dans le modèle standard et dans le modèle d'oracle aléatoire. Nous donnons une description formelle de leur schéma dans le modèle de l'oracle aléatoire qui est basé sur la technique d'échantillonnage généralisé de pré-image (voir les sections 1.5.3.2 et 1.5.3.1 pour plus de détails).

En 2010, Cayrel et al. [CLRS10b] ont proposé un schéma de signature de cercle à seuil dans le modèle de l'oracle aléatoire. Leur analyse de sécurité ne suit pas le modèle de sécurité d'un schéma de signature de cercle. Dans le chapitre 3, nous revenons vers ce résultat que nous considérons plutôt comme étant un schéma d'identification qui permet l'identification anonyme d'un ou plusieurs utilisateurs.

2.2.1 Schéma de Brakerski et Kalai

Soit k le paramètre de sécurité tel que $\mathcal{T}_k$ est un élément de la famille $\mathcal{T}$ définie dans la section 1.5.3.3. Donc, nous avons que $\mathcal{T}_k = \{f : X_k \to Y_k\}$ avec $X_k = \{\boldsymbol{x} \in \mathbb{Z}_q^m : \|\boldsymbol{x}\|_2 \leq s \cdot \sqrt{m}\}$ et $Y_k = \mathbb{Z}_q^k$, où $q = \mathsf{poly}(k), m = (5+\delta)k \log q$ (pour un entier $\delta > 0$), $s = L \cdot \omega(\sqrt{\log k})$, et $L = O(\sqrt{k \log q})$. Une fonction $f_{\boldsymbol{A}} \in \mathcal{T}_k$ est définie par une matrice $\boldsymbol{A}$ telle que $f_{\boldsymbol{A}}(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x} \bmod q$. Nous rappelons que $f_{\boldsymbol{A}}$ peut être définie avec ou sans une trappe, c'est-à-dire, que $\boldsymbol{A}$ peut être choisie de manière uniformément aléatoire dans $\mathbb{Z}_q^{k \times m}$ ou bien elle est générée grâce à l'algorithme TrapGen avec une trappe qui correspond à une matrice $\boldsymbol{S}$. Pour simplifier les notations, nous notons une fonction dans $\mathcal{T}_k$ par f sans mettre la matrice en indice.

Soit $\mu \in \{0,1\}^p$ un message à signer, tel que p est polynomial en k, nous utilisons la notation $\mu = [\mu_1 \| \cdots \| \mu_p]$ tel que $\forall i \in [p], \mu_i \in \{0,1\}$. Nous considérons un cercle formé par ℓ utilisateurs (ℓ est polynomial en k), que nous notons par $R = \{pk_i\}_{i \in [\ell]}$, où chaque utilisateur possède une paire de clés (pk_i, sk_i). La clé pu-

blique pk_i consiste en un ensemble de $2p+1$ fonctions $f_0^{(i)}, (f_{j,b}^{(i)})_{(j,b)\in[p]\times\{0,1\}} \in \mathcal{T}_k$ et un vecteur $\boldsymbol{y}^{(i)} \in \mathbb{Z}_q^k$, telle que parmi toutes les fonctions choisies seule la fonction $f_0^{(i)}$ est générée avec une trappe (une matrice) que nous notons par $\boldsymbol{S}_0^{(i)}$. La clé secrète sk_i consiste en la matrice $\boldsymbol{S}_0^{(i)}$.

$\mathsf{Ring-gen-keys}(1^k)$:

1. Soit $f_0 \in \mathcal{T}_k$, telle que f_0 est générée à l'aide de l'algorithme TrapGen.
2. Soit $\boldsymbol{S}_0$ la trappe de la fonction f_0.
3. Soient $(f_{j,b})_{(j,b)\in[p]\times\{0,1\}} \in \mathcal{T}_k$ des fonctions générées sans trappes.
4. Soit $\boldsymbol{y} \leftarrow \mathbb{Z}_q^k$.
5. Retourner $(pk = (f_0, (f_{j,b})_{(j,b)\in[p]\times\{0,1\}}, \boldsymbol{y}), sk = \boldsymbol{S}_0)$.

$\mathsf{Ring-sign}(sk, \mu, R)$: L'algorithme de signature prend comme entrée un message à signer $\mu = [\mu_1 \| \cdots \| \mu_p]$, un cercle $R = \{f_0^{(i)}, (f_{j,b}^{(i)})_{(j,b)\in[p]\times\{0,1\}}, \boldsymbol{y}^{(i)}\}_{i\in[\ell]}$ et la clé secrète $sk = \boldsymbol{S}_0^{(\tau)}$ ($\tau \in [\ell]$) associée à la clé publique $\{f_0^{(\tau)}, (f_{j,b}^{(\tau)})_{(j,b)\in[p]\times\{0,1\}}, \boldsymbol{y}^{(\tau)}\} \in R$. Une signature est générée de la manière suivante :

1. Soit $\boldsymbol{y} = \boldsymbol{y}^{(1)}$.
2. Choisir $(f_0^{(i)}, f_{j,\mu_j}^{(i)})_{j\in[p],i\in[\ell]}$ à partir de R.
3. En utilisant la clé secrète $\boldsymbol{S}_0^{(\tau)}$, générer $(\boldsymbol{x}_j^{(i)})_{i\in[\ell],j\in\{0\}\cup[p]} \in X_k$ tel que

$$\sum_{i\in[\ell]} f_0^{(i)}(\boldsymbol{x}_0^{(i)}) + \sum_{i\in[\ell],j\in[p]} f_{j,\mu_j}^{(i)}(\boldsymbol{x}_j^{(i)}) = \boldsymbol{y} \bmod q.$$

4. Retourner $\sigma = (\boldsymbol{x}_j^{(i)})_{i\in[\ell],j\in\{0\}\cup[p]}$ comme signature du message μ.

$\mathsf{Ring-verify}(\mu, R, \sigma)$: Le vérificateur accepte la signature si les deux conditions suivantes sont satisfaites :
— $\boldsymbol{x}_j^{(i)} \in X_k$ pour tous $i \in [\ell], j \in \{0\} \cup [p]$.
— $\sum_{i\in[\ell]} f_0^{(i)}(\boldsymbol{x}_0^{(i)}) + \sum_{i\in[\ell],j\in[p]} f_{j,\mu_j}^{(i)}(\boldsymbol{x}_j^{(i)}) = \boldsymbol{y} \bmod q$.

Remarque 2.1. *Dans l'étape* (1) *le signataire et le vérificateur doivent faire un choix consensuel sur un* $\boldsymbol{y}$ *parmi* $\{\boldsymbol{y}^{(i)}\}_{i\in[\ell]}$ *avant la génération de la signature. Par exemple, dans notre description nous avons choisi* $\boldsymbol{y} = \boldsymbol{y}^{(1)}$.

Dans l'étape (3)*, le signataire qui connait la clé secrète* $\boldsymbol{S}_0^{(\tau)}$ *peut générer des vecteurs dans* X_k *qui vérifient l'équation. En effet, ceci est possible d'après la propriété de trappe des fonctions à trappe de cercle (voir la section 1.5.3.3).*

Sécurité. L'anonymat du schéma de Brakerski et Kalai, repose sur la propriété de trappe des fonctions à trappe de cercle. En effet, cette propriété garantit que quelque soit la clé secrète utilisée, les vecteurs qui représentent la signature ont la même distribution.

Dans [BK10], Brakerski et Kalai ont donné une preuve d'inforgeabilité contre les attaques à sous-cercles choisis. Ils ont lié l'inforgeabilité du schéma à la propriété sens-unique des fonctions à trappe de cercle (Proposition 1.8). En effet, un message $\mu \in \{0,1\}^p$ suivant lequel l'attaquant va essayer de forger une signature définit un ensemble de $(p+1)\ell$ fonctions dans $\mathcal{T}_k$ (ℓ désigne la taille du cercle dans la réponse de l'attaquant). Pour obtenir une signature valide de μ, l'attaquant doit trouver des vecteurs qui vérifient les deux conditions de l'algorithme de vérification. Par conséquent ceci revient à inverser un ensemble de fonctions à trappe sans connaitre aucune trappe valide. D'après la propriété de sens-unique des fonctions à trappe de cercle, ceci est aussi difficile que résoudre le problème ISIS dans le cas moyen.

2.2.2 Schéma de Wang et Sun

Soient n le paramètre de sécurité et ℓ un entier polynomial en n. Nous considérons un cercle formé par ℓ utilisateurs, que nous notons par $R = \{pk_i\}_{i\in[\ell]}$, où chaque utilisateur possède une paire de clés (pk_i, sk_i). Chaque paire de clés (pk_i, sk_i) est obtenue grâce à l'algorithme TrapGen (voir la section 1.5.3.1 pour plus de détails), la clé publique pk_i consiste en la matrice $\boldsymbol{A}_i$ et la clé secrète sk_i est la matrice $\boldsymbol{S}_i$ qui représente une trappe pour la fonction définie par la matrice $\boldsymbol{A}_i$. Pour signer un message $\mu \in \{0,1\}^\star$, le signataire utilise les clés publiques (matrices publiques) des utilisateurs du cercle et il construit une matrice $\boldsymbol{A}_R = [\boldsymbol{A}_1 \| \cdots \| \boldsymbol{A}_\ell]$. Ensuite, il fait un appel à un oracle aléatoire H pour obtenir $\boldsymbol{y} = H(\mu)$. En utilisant l'algorithme GenSamplePre (voir la section 1.5.3.2), le signataire génère un vecteur $\boldsymbol{e}$ tel que $\boldsymbol{A}_R\boldsymbol{e} = \boldsymbol{y} \bmod q$, c'est la signature du message μ.

Plus formellement, le schéma proposé par Wang et Sun [WS11] se présente de la manière suivante :

Ring$-$gen$-$params(1^n) : Étant donné le paramètre de sécurité n.

1. Soient m, q des entiers tels que $q = \mathsf{poly}(n)$ et $m \geq 5n \log q$.
2. Soit $s \geq L \cdot \omega(\sqrt{\log n})$ où $L \geq O(\sqrt{n \log q})$.
3. Soit $H_1 : \{0,1\}^\star \to \mathbb{Z}_q^n$, un oracle aléatoire.
4. Retourner $\mathcal{P} = (n, q, m, s, L, H_1)$.

Ring$-$gen$-$keys$(\mathcal{P})$:

1. Soit $(\boldsymbol{A}, \boldsymbol{S}) \leftarrow \mathsf{TrapGen}(1^n)$.
2. Retourner $(pk = \boldsymbol{A}, sk = \boldsymbol{S})$.

Ring$-$sign$(\mathcal{P}, sk, \mu, R)$: Soient un message $\mu \in \{0,1\}^\star$, un cercle $R = \{\boldsymbol{A}_i\}_{i\in[\ell]}$, $sk = \boldsymbol{S}_j$ la clé secrète associée à la clé publique $\boldsymbol{A}_j \in R$.

1. Soit $\boldsymbol{A}_R = [\boldsymbol{A}_1 \| \cdots \| \boldsymbol{A}_\ell] \in \mathbb{Z}_q^{n \times \ell m}$.

2. Soit $\boldsymbol{y} \leftarrow H_1(\mu)$.
3. $\boldsymbol{e} \leftarrow \mathsf{GenSamplePre}(\boldsymbol{A}_R, \boldsymbol{A}_j, \boldsymbol{S}_j, \boldsymbol{y}, s)$.
4. Retourner $\sigma = \boldsymbol{e}$, comme signature du message μ.

$\mathsf{Ring-verify}(\mathcal{P}, \mu, R, \sigma)$: Le vérificateur accepte la signature si les deux conditions suivantes sont satisfaites :

1. $0 \leq \|\boldsymbol{e}\|_2 \leq s \cdot \sqrt{\ell m}$.
2. $\boldsymbol{A}_R \boldsymbol{e} \bmod q = H_1(\mu)$.

Sécurité. Wang et Sun ont démontré que ce schéma vérifie la propriété d'anonymat contre la révélation totale des clés secrètes. Soit R un cercle de taille ℓ tel que $R = \{\boldsymbol{A}_i\}_{i \in [\ell]}$. Nous rappelons que pendant la phase de challenge du jeu d'anonymat (voir Définition 2.5), l'attaquant envoie $(\mu^\star, S^\star, i_0, i_1)$ comme challenge tel que $(i_0, i_1) \in index(S^\star) \cap index(R)$. Ensuite une signature challenge $\sigma_b = \boldsymbol{e}_b \leftarrow \mathsf{Ring-sign}(\mathcal{P}, sk_{i_b}, \mu^\star, S^\star)$ est envoyée à l'attaquant.

D'après la Proposition 1.7, une clé secrète $sk_{i_j} = \boldsymbol{S}_j$, permet à l'algorithme de signature de choisir un vecteur $\boldsymbol{e}_j$ à partir d'une distribution qui est à une distance statistique négligeable de la distribution $\mathsf{D}_{\mathbb{Z}^{\ell m}, s}$. Par conséquent la distance statistique entre $\boldsymbol{e}_0$ et $\boldsymbol{e}_1$ est négligeable, ceci implique que l'avantage de l'attaquant dans le jeu d'anonymat est négligeable.

En ce qui concerne l'inforgeabilité, Wang et Sun ont pontré que leur schéma satisfait l'inforgeabilité contre corruption interne. La proposition suivante montre que forger des signatures suivant ce modèle de sécurité est aussi difficile que le problème SIS.

Proposition 2.1 (Théorème 2 dans [WS11])**.** *S'il existe un algorithme en temps polynomial qui peut gagner le jeu d'inforgeabilité contre corruption interne, alors il existe un algorithme qui peut résoudre le problème* $\mathsf{SIS}_{n,q,\ell m,2s}$ *dans le cas moyen en un temps polynomial.*

Nous renvoyons le lecteur à [WS11] pour le détail de la preuve de sécurité.

2.3 Nouveau schéma de signature de cercle

Dans cette section, nous décrivons notre schéma de signature de cercle.

2.3.1 Description du schéma

Tout d'abord, nous allons montrer comment modifier le schéma de signature LSig09 pour obtenir notre signature de cercle. Ensuite, nous donnons une description plus formelle de notre schéma avec des détails sur le temps d'exécution

ainsi que le coût des opérations pour les algorithmes de génération des clés, de signature et de vérification.

Il existe deux modifications majeures. La première modification est pour assurer que chaque utilisateur a dans sa clé publique une fonction de hachage h_i telle que $h_i(\hat{\mathbf{s}}_i) = \mathbf{S}$, où $\hat{\mathbf{s}}_i$ est la clé secrète et $\mathbf{S}$ un polynôme (non nul) fixé. On considère un cercle $R = \{h_i\}_{i\in[\ell]}$. La deuxième modification est pour garder le signataire anonyme quand il signe des messages. Nous le faisons en rajoutant $\ell - 1$ variables aléatoires qui vont correspondre au membres du cercle sauf le signataire. Par exemple, supposons que le signataire a $j \in [\ell]$ comme indice, le signataire envoie $(\hat{\mathbf{z}}_i; i \in [\ell], \mathbf{e})$ comme signature telle que $\mathbf{e} = H(\sum_{i\in[\ell]} h_i(\hat{\mathbf{y}}_i), \mu)$, $\hat{\mathbf{z}}_j = \hat{\mathbf{s}}_j\mathbf{e} + \hat{\mathbf{y}}_j$ et $\hat{\mathbf{z}}_i = \hat{\mathbf{y}}_i$ pour $i \in [\ell] \setminus \{j\}$. Par conséquent, la signature est constituée de $[\ell]$ éléments, un pour chaque membre du cercle. Maintenant, nous revenons à la première modification et nous montrerons son utilité dans la consistance du schéma. Dans l'étape de vérification le vérificateur vérifie si le haché de $(\sum_{i\in[\ell]} h_i(\hat{\mathbf{z}}_i) - \mathbf{S}\mathbf{e}, \mu)$ est égal à $\mathbf{e}$. Il est simple de remarquer que cela est le cas si $\sum_{i\in[\ell]} h_i(\hat{\mathbf{z}}_i) - \mathbf{S}\mathbf{e}$ est égal à $\sum_{i\in[\ell]} h_i(\hat{\mathbf{y}}_i)$. En effet, en utilisant la linéarité de h_j nous avons $h_j(\hat{\mathbf{z}}_j) = h_j(\hat{\mathbf{s}}_j\mathbf{e} + \hat{\mathbf{y}}_j) = h_j(\hat{\mathbf{y}}_j) + \mathbf{S}\mathbf{e}$. Puisque tous les membres du cercle ont des paires de clés $(h_i, \hat{\mathbf{s}}_i)$ telles que $h_i(\hat{\mathbf{s}}_i) = \mathbf{S}$, le vérificateur va toujours accepter la signature quand l'un d'eux l'a produite.

Remarque 2.2. *Afin d'avoir l'inforgeablité contre les attaques à sous-cercles choisis, nous devons modifier le schéma de telle sorte que la requête pour l'oracle aléatoire contiendra une description du cercle pour lequel la signature est valide. Dans le but de résister aux attaques avec des paramètres choisis par l'attaquant, l'algorithme de signature commence par une étape dans laquelle les entrées doivent passer quelques tests.*

Pour les paramètres donnés dans la Table 2.1, nous considérons la famille de fonctions de hachage $\mathcal{H}(\mathcal{D}, D_h, m)$ comme la famille définie dans la section 1.5.2.2 ($D_\times = D_h$), nous notons par $\mathsf{Col}(h, D_h)$ le problème de collision (voir la Définition 1.33) associé à cette famille. De même nous considérons la famille de fonctions de hachage $\mathcal{H}(\mathcal{D}, D_h, m_u)$ définie par m_u colonnes au lieu de m.

Théorème 2.1 (Théorème 1.3 adapté à nos paramètres)**.** *Soient $\mathcal{D}, n, p$ et m comme dans la Table 2.1. S'il existe un algorithme en temps polynomial qui peut résoudre le problème $\mathsf{Col}(h, D_h)$ pour $h \in \mathcal{H}(\mathcal{D}, D_h, m)$ avec une probabilité non négligeable, alors il existe un algorithme en temps polynomial qui peut résoudre $(x^n + 1)$-$\mathsf{SVP}_\gamma^\infty$ dans le pire cas pour $\gamma = O(n^{2.5+2c})$.*

Remarque 2.3. *Nous choisissons les paramètres et les ensemble nécessaires pour le Théorème 1.3 de la manière suivante ; $D_\times = D_h$, $d = 16(mn^{1.5}\log n + \sqrt{n}\log n)n\log^2 n$ avec n, p et m comme dans la Table 2.1, $\mathbf{f} = x^n + 1$. D'après le Lemme 1.4, nous avons que $\theta(\mathbf{f}) = 1$.*

n	puissance de 2 plus grande que le paramètre de sécurité k
c	une constante positive
p	premier de l'ordre de $\Theta(n^{4+c})$ tel que $p \equiv 3 \bmod 8$
m_u	$(3 + 2c/3) \log n$
m	$(3 + 2c/3)n^c \log n$
$\mathcal{D}$	l'anneau $\mathbb{Z}_p[x]/\langle x^n + 1\rangle$
D_h	$\{\mathbf{g} \in \mathcal{D} : \|\mathbf{g}\|_\infty \leq mn^{1.5} \log n + \sqrt{n} \log n\}$
D_y	$\{\mathbf{g} \in \mathcal{D} : \|\mathbf{g}\|_\infty \leq mn^{1.5} \log n\}$
D_z	$\{\mathbf{g} \in \mathcal{D} : \|\mathbf{g}\|_\infty \leq mn^{1.5} \log n - \sqrt{n} \log n\}$
$D_{s,c}$	$\{\mathbf{g} \in \mathcal{D} : \|\mathbf{g}\|_\infty \leq 1\}$
H	un oracle aléatoire $H : \{0,1\}^\star \to D_{s,c}$

TABLE 2.1 – Ensembles et paramètres.

Dans le schéma de signature de cercle que nous allons présenter, les fonctions de hachage que nous manipulons seront toujours dans la famille $\mathcal{H}(\mathcal{D}, D_h, m')$ avec $m' \leq m$. Il est important de remarquer que si un attaquant est capable de résoudre le problème de collision (Définition 1.33) pour $m' \leq m$, alors il peut aussi le résoudre pour m. En effet, étant donnée une fonction $h \in \mathcal{H}(\mathcal{D}, D_h, m)$, l'attaquant peut extraire un vecteur de m' polynômes à partir du vecteur de polynômes de h. Ensuite, il résout le problème de collision pour m' et pour obtenir une collision pour h, il complète les cordonnées qui manquent par des zéros.

$\mathsf{Ring{-}gen{-}params}(1^k)$: Étant donné le paramètre de sécurité k.

1. Soit n une puissance de deux et plus grand que k.
2. Soient $m_u = (3 + 2c/3) \log n$, et p premier plus grand que n^4 tel que $p = 3 \bmod 8$.

— Remarque : ces paramètres définissent les ensembles $\mathcal{D}, D_h, D_z, D_y, D_{s,c}$ et la famille $\mathcal{H}(\mathcal{D}, D_h, m_u)$.

3. Soit $\mathbf{S} \leftarrow \mathcal{D}$, $\mathbf{S} \neq 0$
4. Retourner $\mathcal{P} = (k, n, m_u, p, \mathbf{S})$

$\mathsf{Ring{-}gen{-}keys}(\mathcal{P})$

1. Soit $\hat{\mathbf{s}} = (\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_{m_u}) \leftarrow D_{s,c}^{m_u}$
2. Si aucun des $\mathbf{s}_i$ est inversible, revenir à 1.
3. Soit $i_0 \in \{1, \ldots, m\}$ tel que $\mathbf{s}_{i_0}$ soit inversible.
4. Soit $(\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_{i_0-1}, \mathbf{a}_{i_0+1}, \ldots, \mathbf{a}_{m_u}) \leftarrow \mathcal{D}^{m_u-1}$.
5. Soit $\mathbf{a}_{i_0} = \mathbf{s}_{i_0}^{-1}(\mathbf{S} - \sum_{i \neq i_0} \mathbf{a}_i \mathbf{s}_i)$ et noter $\hat{\mathbf{a}} = (\mathbf{a}_1, \ldots, \mathbf{a}_{m_u})$.
6. Retourner $(pk, sk) = (h, \hat{\mathbf{s}})$, h étant la fonction de hachage dans $\mathcal{H}(\mathcal{D}, D_h, m_u)$ définie par $\hat{\mathbf{a}}$.

Ring$-$sign$(\mathcal{P}, sk, \mu, R)$: Soient $\mu \in \{0,1\}^\star$ un message, un cercle $R = \{h_i\}_{i\in[\ell]}$ de ℓ membres (ℓ est un entier polynomial en k) tel que $\{h_i\}_{i\in[\ell]} \subset \mathcal{H}(\mathcal{D}, D_h, m_u)$, et $sk = \hat{\mathbf{s}}_j$ la clé secrète associée à une clé publique h_j dans R.

0. Vérifier que : les paramètres publics satisfont les contraintes $1-3$ dans Ring$-$gen$-$params ; sk est dans $D_{s,c}^{m_u}$; R est de taille majorée par k^c ; l'une des clés publiques dans R est associée à sk. Si la vérification échoue retourner `failed`.
1. Pour tout $i \in [\ell]$; $i \neq j$;$\hat{\mathbf{y}}_i \leftarrow D_z^{m_u}$
2. Pour $i = j$; $\hat{\mathbf{y}}_j \leftarrow D_y^{m_u}$
3. Soit $\mathbf{e} \leftarrow H(\sum_{i\in[\ell]} h_i(\hat{\mathbf{y}}_i), R, \mu)$ ($\mathbf{e}$ est donc dans $D_{s,c}$)
4. Pour $i = j$, $\hat{\mathbf{z}}_j \leftarrow \hat{\mathbf{s}}_j\mathbf{e} + \hat{\mathbf{y}}_j$
5. Si $\hat{\mathbf{z}}_j \notin D_z^{m_u}$, revenir à la deuxième étape
6. Pour $i \neq j$, $\hat{\mathbf{z}}_i = \hat{\mathbf{y}}_i$
7. Retourner $\sigma = (\hat{\mathbf{z}}_i; i \in [\ell], \mathbf{e})$, comme signature du message μ.

Ring$-$verify$(\mathcal{P}, \mu, R, \sigma)$: Étant donnés un message μ, un cercle $R = \{h_i\}_{i\in[\ell]}$ et une signature $\sigma = (\hat{\mathbf{z}}_i; i \in [\ell], \mathbf{e})$, le vérificateur accepte la signature si les conditions suivantes sont satisfaites :

1. $\hat{\mathbf{z}}_i \in D_z^{m_u}$ pour tout $i \in [\ell]$
2. $\mathbf{e} = H(\sum_{i\in[\ell]} h_i(\hat{\mathbf{z}}_i) - \mathbf{S}\mathbf{e}, R, \mu)$

Sinon, le vérificateur refuse la signature.

2.3.2 Consistance et convergence des algorithmes de la signature

Proposition 2.2. *Soit* $\sigma = (\hat{\mathbf{z}}_i; i \in [\ell], \mathbf{e}) \leftarrow$ Ring$-$sign$(\mathcal{P}, \hat{\mathbf{s}}_j, \mu, \{h_i\}_{i\in[\ell]})$ *une signature telle que* $j \in [\ell]$ *et* $(h_j, \hat{\mathbf{s}}_j)$ *est une paire de clés donnée. Une signature valide passe toujours les deux tests de* Ring$-$verify.

Démonstration. Les éléments de l'ensemble $(\hat{\mathbf{z}}_i; i \in [\ell])$ passent avec succès le premier test de l'algorithme de vérification Ring$-$verify. En effet, d'après l'étape (5) de l'algorithme de signature, si $\hat{\mathbf{z}}_j \notin D_z^{m_u}$, alors l'algorithme commence à itérer [2] les étapes $(2), (3), (4), (5)$ jusqu'à obtenir $\hat{\mathbf{z}}_j$ qui appartient à $D_z^{m_u}$. D'autre part d'après l'étape (6), nous avons que $\hat{\mathbf{z}}_i = \hat{\mathbf{y}}_i$ pour $i \neq j$ et d'après l'étape (2) nous avons que $\hat{\mathbf{y}}_i \in D_z^{m_u}$ pour tout $i \neq j$. Donc, nous avons démontré que $\hat{\mathbf{z}}_i \in D_z^{m_u}$ pour tout $i \in [\ell]$.

2. Nous montrons dans la Proposition 2.3 qu'en moyenne le nombre d'itération est au plus égal à 3.

En ce qui concerne le deuxième test nous avons :

$$\begin{aligned}\sum_{i\in[\ell]} h_i(\hat{\mathbf{z}}_i) - \mathbf{Se} &= h_j(\hat{\mathbf{z}}_j) - \mathbf{Se} + \sum_{i\in[\ell]\setminus\{j\}} h_i(\hat{\mathbf{z}}_i) \\ &= h_j(\hat{\mathbf{s}}_j\mathbf{e} + \hat{\mathbf{y}}_j) - \mathbf{Se} + \sum_{i\in[\ell]\setminus\{j\}} h_i(\hat{\mathbf{y}}_i)\end{aligned}$$

En remplaçant $\hat{\mathbf{z}}_j$ par $\hat{\mathbf{s}}_j\mathbf{e} + \hat{\mathbf{y}}_j$ et $\hat{\mathbf{z}}_i$ par $\hat{\mathbf{y}}_i$, nous obtenons

$$\sum_{i\in[\ell]} h_i(\hat{\mathbf{z}}_i) - \mathbf{Se} = h_j(\hat{\mathbf{s}}_j)\mathbf{e} + h_j(\hat{\mathbf{y}}_j) - \mathbf{Se} + \sum_{i\in[\ell]\setminus\{j\}} h_i(\hat{\mathbf{y}}_i)$$

En utilisant les propriétés homomorphes de $h_j \in \mathcal{H}(\mathcal{D}, D_h, m_u)$,

$$\sum_{i\in[\ell]} h_i(\hat{\mathbf{z}}_i) - \mathbf{Se} = \sum_{i\in[\ell]} h_i(\hat{\mathbf{y}}_i) \qquad \text{puisque } h_j(\hat{\mathbf{s}}_j) = \mathbf{S}.$$

Comme $\mathbf{e} = H(\sum_{i\in[\ell]} h_i(\hat{\mathbf{y}}_i), \{h_i\}_{i\in[\ell]}, \mu)$, le deuxième test de Ring−verify est toujours vrai pour une signature valide. □

Pour étudier les performances de notre schémas nous aurons besoin de quelques lemmes.

Lemme 2.1 (Lemme 3 dans [SSTX09])**.** *Soient* $\mathbf{f} = x^n + 1$ *et* $n = 2^r$ *pour* $r \geq 2$*. Si* p *est premier tel que* $p \equiv 3 \bmod 8$*, alors il existe deux polynômes* $\mathbf{f}_1, \mathbf{f}_2$ *tels que* $\mathbf{f} = \mathbf{f}_1\mathbf{f}_2 \bmod p$*, où chaque* $\mathbf{f}_i$ *est irréductible dans* $\mathbb{Z}_p[x]$ *et peut s'écrire* $\mathbf{f}_i(x) = x^{n/2} + t_i x^{n/4} - 1$ *pour* $t_i \in \mathbb{Z}_p$*.*

Lemme 2.2. *Soit* $D_{s,c}^{\times}$ *l'ensemble des polynômes non inversible de* $D_{s,c}$*. Nous avons*

$$\mathsf{Pr}_{\mathbf{f}\leftarrow D_{s,c}}\left[\mathbf{f} \in D_{s,c}^{\times}\right] \leq \frac{2}{3^{n/2}}.$$

Démonstration. On a $\mathcal{D} = \mathbb{Z}_p[x]/\langle x^n + 1\rangle$, par le Lemme 2.1 et la Table 2.1, nous avons $x^n + 1 = \mathbf{f}_1\mathbf{f}_2 \bmod p$, où les deux facteurs $\mathbf{f}_1$ et $\mathbf{f}_2$ sont irréductibles dans $\mathbb{Z}_p[x]$ et de degré égal à $n/2$. Comme ces facteurs sont irréductibles, alors les polynômes non inversible de $\mathcal{D}$ sont tels que $\mathbf{f} = 0 \bmod \mathbf{f}_1$ ou bien $\mathbf{f} = 0 \bmod \mathbf{f}_2$.
Pour $i \in \{1, 2\}$, soit

$$F_i = \{\mathbf{g} \in \mathbb{Z}_p[x]/\langle x^n + 1\rangle : \mathbf{f}_i \text{ divise } \mathbf{g}\}$$

qui est de dimension $n/2$ sur $\mathbb{Z}_p$.
Puisque $D_{s,c} = \{\mathbf{g} \in \mathbb{Z}_p[x]/\langle x^n + 1\rangle : \|\mathbf{g}\|_\infty \leq 1\}$ nous avons alors :

$$D_{s,c}^{\times} = (D_{s,c} \cap F_1) \cup (D_{s,c} \cap F_2).$$

En utilisant l'inégalité de Boole, nous obtenons :

$$\mathsf{Pr}_{\mathbf{f}\leftarrow D_{s,c}}\left[\mathbf{f}\in D_{s,c}^{\times}\right]\leq \mathsf{Pr}_{\mathbf{f}\leftarrow D_{s,c}}\left[\mathbf{f}\in (D_{s,c}\cap F_1)\right]+\mathsf{Pr}_{\mathbf{f}\leftarrow D_{s,c}}\left[\mathbf{f}\in (D_{s,c}\cap F_2)\right].$$

Maintenant, nous allons calculer une majoration du cardinal de l'ensemble $(D_{s,c}\cap F_1)$. Tout d'abord, nous définissons la forme linéaire suivante. Soit $\mathbf{w}\in\mathbb{Z}_p[x]$ tel que $\mathbf{w}=\sum_{k=1}^{n}a_{k-1}x^{k-1}$, pour $i\in\{n\}$, $L_i(\mathbf{w})$ est une forme linéaire définie comme suit

$$\begin{array}{rccc} L_i & : & \mathbb{Z}_p[x]/\langle x^n+1\rangle & \rightarrow & \mathbb{Z}_p \\ & & \mathbf{w} & \mapsto & a_{i-1}\end{array}$$

Nous pouvons écrire $D_{s,c}\cap F_1=\{\mathbf{g}\in F_1 : L_1(\mathbf{g}),L_2(\mathbf{g}),\cdots,L_n(\mathbf{g})\in\{-1,0,1\}\}$. Il est clair que la famille $(L_i)_{i\in[n]}$ engendre $(\mathbb{Z}_p[x]/\langle x^n+1\rangle)^\star$ l'espace dual de $\mathbb{Z}_p[x]//\langle x^n+1\rangle$. De même, nous avons que la restriction de famille $(L_i)_{i\in[n]}$ à l'ensemble F_1 noté par $(L_i)_{i\in[n]|_{F_1}}$ engendre $F_1^\star$ (le dual de F_1). Puisque $F_1^\star$ a la même dimension que F_1, c'est-à-dire $n/2$, alors nous pouvons extraire une base de $F_1^\star$. Nous notons $(L_{i_t})_{1\leq t\leq n/2}$ cette base.

Nous obtenons que,

$$D_{s,c}\cap F_1\subset\{\mathbf{g}\in F_1 : \forall 1\leq t\leq n/2, L_{i_t}(\mathbf{g})\in\{-1,0,1\}\}.$$

Nous montrerons que le cardinal de l'ensemble $G:=\{\mathbf{g}\in F_1 : \forall 1\leq t\leq n/2, L_{i_t}(\mathbf{g})\in\{-1,0,1\}\}$ est égal à $3^{n/2}$.

Soit $B=\{\mathbf{b}_1,\cdots,\mathbf{b}_{n/2}\}$ une base de F_1, donc un polynôme $\mathbf{g}\in F_1$ peut s'écrire $\mathbf{g}=\sum_i\alpha_i\mathbf{b}_i$ tel que $\forall i,\alpha_i\in\mathbb{Z}_p$. Si $\mathbf{g}\in G$ alors nous avons $L_{i_t}(\sum_i\alpha_i\mathbf{b}_i)\in\{-1,0,1\},\forall t\in\{1,\cdots,n/2\}$. Par linéarité nous obtenons les $n/2$ équations suivantes tel que $a_i\in\{-1,0,1\}$ pour $i\in\{1,\cdots,n/2\}$.

Nous fixons des valeurs quelconques dans $\{-1,0,1\}$ pour chaque $a_i; i\in\{1,\ldots,n/2\}$. Soit $\mathbf{g}\in F_1$, nous obtenons les $n/2$ équations suivantes à $n/2$ inconnus (*i.e.* les α_i).

$$(\sum_i\alpha_iL_{i_1}(\mathbf{b}_i)=a_1,\cdots,\sum_i\alpha_iL_{i_{n/2}}(\mathbf{b}_i)=a_{n/2})$$

Ce système d'équations admet une solution unique $(\alpha_1,\ldots,\alpha_{n/2})$, par conséquent nous avons trouvé un élément $\mathbf{g}\in G$. Comme il y a $3^{n/2}$ systèmes d'équation nous obtenons que $|G|=3^{n/2}$.

Pour finir la preuve, nous allons montrer que le système d'équations admet toujours une solution unique.
Soit la matrice $\boldsymbol{M}=(L_i(\mathbf{b}_j))$, nous allons montrer que l'application linéaire de matrice $\boldsymbol{M}$ est bijective. Soit $\boldsymbol{x}=(x_1,\cdots,x_{n/2})$ tel que $\boldsymbol{M}\boldsymbol{x}=0$, alors on a

$$\sum_{j=1}^{n/2}x_jL_j(\mathbf{b}_i)=0,\forall i\in\{1,\cdots,n/2\}.$$

Soit $L=\sum_{j=1}^{n/2}x_jL_j$, alors on a $L(\mathbf{b}_i)=0,\forall i\in\{1,\cdots,n/2\}$. Ce qui implique que $L(F_1)=0$ et par conséquent on a $L=0$.

Donc $x_i = 0, \forall i \in \{1, \cdots, n/2\}$ car $(L_i)_{1 \leq i \leq n/2}$ est une base. Ce qui nous permet de conclure que l'application linéaire de la matrice $\boldsymbol{M}$ est injective et donc bijective.

Conclusion : Pour $i \in \{1, 2\}$ nous avons $\mathsf{Pr}_{\mathbf{f} \leftarrow D_{s,c}}[\mathbf{f} \in (D_{s,c} \cap F_i)] \leq \frac{1}{3^{n/2}}$ et ceci termine la preuve. □

Lemme 2.3 (Corollaire 6.2 dans [Lyu08b]). *Pour tout* $\hat{\mathbf{s}} \in D_{s,c}^{m_u}$,

$$\mathsf{Pr}_{\mathbf{c} \leftarrow D_{s,c}, \hat{\mathbf{y}} \leftarrow D_y^{m_u}}[\hat{\mathbf{s}}\mathbf{c} + \hat{\mathbf{y}} \in D_z^{m_u}] = \frac{1}{e} - o(1).$$

Proposition 2.3. *L'espérance de la durée d'exécution des algorithmes* Ring−gen−params, Ring−gen−keys, Ring−sign *et* Ring−verify *est polynomiale en le paramètre de sécurité.*

Démonstration. La première étape de l'algorithme de génération des clés Ring−gen−keys consiste à choisir $\hat{\mathbf{s}}$ de $D_{s,c}^{m_u}$. Ceci correspond à choisir $m_u n = (3 + 2c/3)n \log n$ nombres d'une manière aléatoire de l'ensemble $\{-1, 0, 1\}$. Le Lemme 2.2 montre qu'un polynôme choisi dans la première étape de l'algorithme de génération de clés est inversible avec une probabilité proche de un. Dans la quatrième étape, il faut choisir $(m_u - 1)n$ nombres aléatoire de l'ensemble $\{-\frac{p-1}{2}, \cdots, 0, \cdots, \frac{p-1}{2}\}$. En utilisant le Transformée de Fourier rapide, le calcul de $\mathbf{a}_{i_0}$ dans la cinquième étape coûte $\tilde{O}(n)$ en temps. Par conséquent, le temps d'exécution de l'algorithme Ring−gen est polynomial.

Dans les deux premières étapes, l'algorithme de signature Ring−sign génère $\hat{\mathbf{y}}_i \in D_z^{m_u}$ pour $i \in [\ell]; i \neq j$ et $\hat{\mathbf{y}}_j \in D_y^{m_u}$. Puis il calcule $\sum_{i \in [\ell]} h_i(\hat{\mathbf{y}}_i)$, il fait un appel à l'oracle aléatoire H et il calcule $\hat{\mathbf{z}}_j = \hat{\mathbf{s}}_j \mathbf{e} + \hat{\mathbf{y}}_j$. Le coût de toutes ces étapes est $\tilde{O}(n)$. Maintenant, il reste à estimer le nombre d'itérations parce que si $\hat{\mathbf{z}}_j \notin D_z^{m_u}$, on devrait aller à la deuxième étape de l'algorithme. D'après le Lemme 2.3 $\hat{\mathbf{z}}_j$ va être dans l'ensemble $D_z^{m_u}$ avec une probabilité de $1/e - o(1)$. Ce qui implique que le nombre prévu d'itérations dans Ring−sign est plus petit que 3. Donc le temps d'exécution de Ring−sign est $\tilde{O}(n)$.

L'algorithme de vérification calcule $\sum_{i \in [\ell]} h_i(\hat{\mathbf{z}}_i)$, Se, puis il soustrait ces deux quantités et ensuite il fait un appel à l'oracle aléatoire H. Ces opérations prennent un temps en $\tilde{O}(n)$.

□

2.3.3 Distribution de la clé publique

Dans cette section nous montrons que les clés publiques obtenues grâce à l'algorithme de génération des clés Ring−gen−keys ont une distribution statistiquement proche de la distribution uniforme sur $\mathcal{D}$. Afin de prouver ce résultat nous aurons besoin du lemme suivant :

Lemme 2.4 (Lemme 6 dans [SSTX09]). *Soient $\mathbb{F}$ un corps fini et $\mathbf{f} \in \mathbb{F}[x]$ un polynôme unitaire de degré $n > 0$. Soient A l'anneau $\mathbb{F}/\mathbf{f}$, $S \subset \mathbb{F}$ et $r > 0$. Pour $\mathbf{a}_1, \cdots, \mathbf{a}_r \in A$, soit $H(\mathbf{a}_1, \cdots, \mathbf{a}_r)$ la variable aléatoire qui représente $\sum_{i \leq r} b_i \mathbf{a}_i \in A$, où les b_i sont des polynômes de degré inférieur à n de coefficients choisis uniformément dans S. Soient $U_1, \cdots, U_{r+1}$ des variables aléatoires indépendantes choisies de manière uniformément aléatoire dans A, alors nous avons que*

$$\Delta((U_1, \cdots, U_r, H(U_1, \cdots, U_r)), (U_1, \cdots, U_{r+1})) \leq \frac{1}{2}\sqrt{\Pi_{i \leq t}\left(1 + \left(\frac{|\mathbb{F}|}{|S|^r}\right)^{deg\ \mathbf{f}_i}\right) - 1},$$

avec $\mathbf{f} = \Pi_{i \leq t}\mathbf{f}_i$ est la factorisation de $\mathbf{f}$ dans $\mathbb{F}$.

Le théorème suivant est une simple adaptation du lemme précédant.

Théorème 2.2. *Pour $\mathbf{g}_1, \ldots, \mathbf{g}_{m_u - 1} \in \mathcal{D}$, on note par $F(\mathbf{g}_1, \ldots, \mathbf{g}_{m_u-1})$ la variable aléatoire $\sum_{i \in [m_u - 1]} \mathbf{s}_i \mathbf{g}_i \in \mathcal{D}$ avec les $\mathbf{s}_1, \ldots, \mathbf{s}_{m_u-1}$ choisis de manière uniformément aléatoire dans $D_{s,c}$. Soient $U_1, \ldots, U_{m_u}$ des variables aléatoires indépendantes choisies d'une manière aléatoire dans D, nous avons*

$$\Delta((U_1, \ldots, U_{m_u-1}, F(U_1, \ldots, U_{m_u-1})), (U_1, \ldots, U_{m_u})) \leq \frac{1}{2}\sqrt{\left(1 + \left(\frac{p}{3^{m_u-1}}\right)^{n/2}\right) - 1}$$

Démonstration. Nous obtenons la démonstration par une simple application du Lemme 2.4, avec nos paramètres. Nous utilisons aussi le Lemme 2.1 qui garantit que $x^n + 1 = \mathbf{f}_1\mathbf{f}_2 \bmod p$ avec le choix de nos paramètres, où chaque $\mathbf{f}_i$ est irréductible dans $\mathbb{Z}_p[x]$ et peut s'écrire $\mathbf{f}_i(x) = x^{n/2} + t_i x^{n/4} - 1$ avec $t_i \in \mathbb{Z}_p$. □

Corollaire 2.1 (Uniformité de la clé publique). *Soit $X_{\mathcal{P}}$ une variable aléatoire qui représente la distribution des fonctions de hachage obtenues grâce à l'algorithme de génération des clés* Ring−gen−keys($\mathcal{P}$). *On considère les variables indépendantes $U_1, \ldots, U_{m_u}$ choisies de manière uniformément aléatoire dans $\mathcal{D}$. Nous avons*

$$\Delta\left(X_{\mathcal{P}}, (U_1, \ldots, U_{m_u})\right) \leq n^{-\omega(1)}.$$

Démonstration. Nous représentons une fonction de hachage h par l'ensemble des polynômes $\hat{\mathbf{a}} = (\mathbf{a}_1, \ldots, \mathbf{a}_{m_u})$. Nous supposons (sans perte de généralité) que $(\mathbf{a}_1, \ldots, \mathbf{a}_{m_u}) = (\mathbf{a}_1, \ldots, \mathbf{a}_{m_u-1}, \mathbf{s}_{m_u}^{-1}(\mathbf{S} - \sum_{i \in [m_u-1]} \mathbf{a}_i \mathbf{s}_i))$, tel que $\hat{\mathbf{s}} = (\mathbf{s}_1, \ldots, \mathbf{s}_{m_u})$ est la clé secrète qui correspond à h.

Tout d'abord, nous remarquons que la fonction à droite de l'inégalité dans le Théorème 2.2 est négligeable pour notre choix de paramètres. En effet, puisque $p = \Theta(n^{4+c})$ et $3^{m_u-1} = 3^{(3+2c/3)\log_2 n}/3 = n^{(3+2c/3)\log_2 3}/3 > n^{4.5+c}/3$. Par conséquent, en utilisant le Théorème 2.2 nous avons,

$$\Delta((U_1, \ldots, U_{m_u-1}, F(U_1, \ldots, U_{m_u-1})), (U_1, \ldots, U_{m_u})) \leq n^{-\omega(1)}.$$

D'après la Proposition 1.3, une fonction ne peut pas augmenter la distance statistique. On considère $f(\mathbf{g}_1, \ldots, \mathbf{g}_{m_u})$, la fonction qui laisse inchangés les $m_u - 1$ premières cordonnées et qui remplace $\mathbf{g}_{m_u}$ par $\mathbf{s}_{m_u}^{-1}(\mathbf{S} - \mathbf{g}_{m_u})$, alors nous obtenons

$$\Delta(f(U_1, \ldots, U_{m_u-1}, F(U_1, \ldots, U_{m_u-1})), f(U_1, \ldots, U_{m_u})) \leq n^{-\omega(1)}.$$

Pour démontrer le corollaire il suffit de remarquer que $f(U_1, \ldots, U_{m_u-1}, F(U_1, \ldots, U_{m_u-1}))$ a exactement la même distribution que $X_{\mathcal{P}}$, et que $f(U_1, \ldots, U_{m_u})$ a la même que $(U_1, \ldots, U_{m_u})$. La première assertion est évidente d'après l'algorithme de génération des clés $\mathsf{Ring-gen-keys}$. La deuxième est dûe au fait que l'addition d'un élément (dans notre cas $\mathbf{S}$) ou bien la multiplication par un élément inversible (dans notre cas $\mathbf{s}_{m_u}^{-1}$) dans $\mathcal{D}$ est une permutation de l'élément de l'anneau et donc la distribution uniforme reste inchangée. Par conséquent nous avons

$$\Delta(f(U_1, \ldots, U_{m_u-1}, F(U_1, \ldots, U_{m_u-1})), f(U_1, \ldots, U_{m_u})) = \Delta(X_{\mathcal{P}}, (U_1, \ldots, U_{m_u}))$$

et donc, $\Delta(X_{\mathcal{P}}, (U_1, \ldots, U_{m_u})) \leq n^{-\omega(1)}$. □

2.4 Analyse de sécurité

Cette section sera consacrée à l'étude de sécurité de notre nouveau schéma. Dans la section 2.4.1, nous montrons que notre schéma assure l'anonymat inconditionnel, au sens de la Définition 2.6. Par conséquent, même un attaquant ayant une puissance de calcul infinie ne peut deviner l'identité de signataire. Ensuite, dans la section 2.4.2, nous donnons une preuve d'inforgeabilité contre les attaques à sous-cercles choisis.

2.4.1 Anonymat

Dans les sections 2.4.1.1 et 2.4.1.2, nous analysons le fait que dans le jeu d'anonymat, un attaquant peut avoir accès à des informations supplémentaires. Nous montrons que dans notre cas ceci ne va pas l'aider pour distinguer les signatures. Ensuite, dans la section 2.4.1.3, nous donnons la preuve d'anonymat pour notre schéma de signature de cercle suivant la notion d'anonymat de la Définition 2.6.

2.4.1.1 Attaquant ayant accès à plusieurs signatures

Dans la preuve d'anonymat de notre schéma, nous modélisons la signature comme étant une variable aléatoire. Pendant le jeu d'anonymat, l'attaquant va essayer de distinguer entre deux signatures ce qui revient à distinguer deux distribution de deux variables aléatoires. Chacune de ces variables est générée en utilisant une clé secrète connue par l'attaquant.

Remarque 2.4. *Soient (X_k) et (X'_k) deux familles de variables aléatoires telles que la distance statistique $\Delta(X_k, X'_k)$ est négligeable. Nous modélisons les moyens algorithmiques de l'attaquant par la fonction f. D'après la Proposition 1.3, l'attaquant aura un avantage négligeable lorsqu'il va essayer de distinguer entre deux distributions des variables de la famille (X_k) et (X'_k).*

De plus, dans la Proposition 1.3 il n'y a pas d'hypothèse sur la complexité computationnelle de f, c'est à dire si elle est computationellement bornée ou pas. Par conséquent, quelque soit la complexité computationelle de l'attaquant, si la distance statistique entre les deux familles est négligeable alors son avantage restera toujours négligeable.

Dans la proposition suivante nous montrons que la distance statistique entre deux variables aléatoires peut augmenter quand on observe plusieurs variables aléatoires par distribution.

Proposition 2.4. *Soient ϕ et ϕ' deux distributions sur un ensemble A. Soient (X, Y) deux variables de ϕ et (X', Y') de ϕ', nous avons*

$$2\Delta(X, X') \geq \Delta((X, Y), (X', Y')) \geq \Delta(X, X'). \tag{2.1}$$

Démonstration. Commençons par montrer que $\Delta((X, Y), (X', Y')) \leq 2\Delta(X, X')$. Par la propriété d'inégalité triangulaire nous avons :

$$\Delta((X, Y), (X', Y')) \leq \Delta((X, Y), (X, Y')) + \Delta((X, Y'), (X', Y')) \tag{2.2}$$

D'après la Proposition 1.1, nous obtenons

$$(2.2) \leq \Delta(X, X') + \Delta(Y, Y') \leq 2\Delta(X, X').$$

Maintenant nous montrons que $\Delta((X, Y), (X', Y')) \geq \Delta(X, X')$. Soit g une fonction telle que $g(X_1, Y_1) = X_1$ pour $(X_1, Y_1) \in A$. Par conséquent d'après la Proposition 1.3, nous avons que

$$\Delta(X, X') = \Delta(g(X, Y), g(X', Y')) \leq \Delta((X, Y), (X', Y')).$$

□

Par conséquent, si un attaquant possède plusieurs éléments de ϕ et de ϕ' alors il peut être capable de mieux distinguer qu'avec un seul élément de chaque distribution. Plus précisément, en utilisant plusieurs fois l'équation (2.1), supposons que l'attaquant possède $\#s$ éléments de chaque distribution tels que la distance statistique de la famille des variables aléatoires est majorée par $\varepsilon(k)$ (k étant le paramètre de sécurité). Alors, nous avons que l'avantage de l'attaquant est majoré par $\#s \cdot \varepsilon(k)$.

Remarque 2.5. *Si $\varepsilon(k) = 2^{-k}$ alors pour un nombre polynomial d'éléments, par exemple $\#s = 2^{k/2}$, l'avantage de l'attaquant continue d'être majoré par une fonction négligeable en le paramètre de sécurité k. En effet nous avons que $\#s \cdot \varepsilon(k) = 2^{-k/2}$.*

2.4.1.2 Attaquant avec plus d'information

Dans le jeu d'anonymat un attaquant peut avoir des informations supplémentaires sur les variables aléatoires qui représentent les deux signatures à distinguer. Par exemple, il peut obtenir les paramètres publics du schéma ainsi que les clés publiques utilisées pendant le jeu.

Soit un attaquant dont les moyens algorithmiques sont modélisés par une fonction f. Supposons qu'il doit distinguer entre les distributions de deux variables aléatoires X_k et X'_k, nous modélisons l'information supplémentaire obtenu par l'attaquant par la variable aléatoire Z_k. Analysons l'utilité de cette information pour l'attaquant. Nous distinguons deux cas :

1. si X_k et X'_k sont indépendantes de Z_k, alors cette information supplémentaire va être inutile pour l'attaquant. En effet, en utilisant la Proposition 1.1, nous avons que $\Delta((X_k, Z_k), (X'_k, Z_k)) = \Delta(X_k, X'_k)$ et par conséquent nous avons $\Delta(f(X_k, Z_k), f(X'_k, Z_k)) \leq \Delta(X_k, X'_k)$.
2. si X_k et X'_k ne sont pas indépendantes de Z_k, alors on ne peut pas utiliser le même argument, puisque la Proposition 1.1 ne peut être appliquée que sur des variables indépendantes. En revanche, en notant $X_{k,z}$ et $X'_{k,z}$ les variables aléatoires quand la variable $Z = z$, si on a

   $$\Delta(X_{k,z}, X'_{k,z}) < \epsilon(k)$$

   qui est indépendante de z, alors $\epsilon(k)$ est une majoration pour $\Delta(f(X_k, Z_k), f(X'_k, Z'_k))$. En effet, nous avons :

$$\begin{aligned}
\Delta(f(X_k, Z_k), f(X'_k, Z_k)) &\leq \Delta((X_k, Z_k), (X'_k, Z_k)) \\
&= \frac{1}{2} \sum_{x,z} |Pr[(X_k, Z_k) = (x, z)] - Pr[(X'_k, Z_k) = (x, z)]| \\
&= \frac{1}{2} \sum_{z} Pr[Z_k = z] \sum_{x} |Pr[X_k = x \mid Z_k = z] - Pr[X'_k = x \mid Z_k = z]| \\
&= \sum_{z} Pr[Z_k = z] \Delta(X_{k,z}, X'_{k,z})
\end{aligned}$$

   Par conséquent,

   $$\Delta(f(X_k, Z_k), f(X'_k, Z_k)) \leq \epsilon(k). \tag{2.3}$$

Remarque 2.6. *Dans notre cas, l'attaquant va essayer de distinguer entre deux variables aléatoires X_k et X'_k qui dépendent d'une variable Z_k. En effet, on est dans la configuration de deuxième cas, puisque les signatures (modélisées ici par X_k et X'_k) dépendent toujours des clés publiques (représentées par la variable Z_k).*

Il est important de remarquer que la majoration dans (2.3) ne dépend pas de la distribution suivie par la variable aléatoire Z_k, ce qui implique qu'on peut inclure dans cette variable des paramètres choisis par l'attaquant sans modifier la distance statistique.

2.4.1.3 Preuve d'anonymat de notre schéma

Dans le jeu d'anonymat inconditionnel contre les attaques à paramètres choisis, l'attaquant reçoit une signature qui dépend d'un bit aléatoire b ainsi qu'un ensemble de paramètres publiques $\mathcal{P} = (k, n, m_u, p, \mathbf{S})$, deux clés secrètes sk_{i_0}, sk_{i_1}, un message μ et un cercle ayant R comme ensemble de clés publiques. Tous ces paramètres sont choisis par l'attaquant sauf le bit aléatoire b.

Soit $X_{b,\mathcal{P},sk_{i_b},\mu,R}$ la variable aléatoire qui représente la signature reçue par l'attaquant pour des paramètres donnés. Le théorème suivant affirme que pour n'importe quel choix de $\mathcal{P}, sk_{i_0}, sk_{i_1}, \mu, R$, la distance statistique entre $X_{0,\mathcal{P},R,sk_{i_0},\mu}$ et $X_{1,\mathcal{P},R,sk_{i_1},\mu}$ est négligeable en le paramètre de sécurité k.

Par conséquent, en utilisant les propriétés de la distance statistique vues dans les sections 2.4.1.1 et 2.4.1.2, ceci implique que notre schéma vérifie la propriété d'anonymat inconditionnel contre les attaques à paramètres choisis. En effet, si on a $\Delta(X_{0,\mathcal{P},sk_{i_0},\mu,R}, X_{1,\mathcal{P},sk_{i_1},\mu,R}) = n^{-\omega(1)} = 2^{-\omega(1)\cdot c}$, d'après 2.4.1.1 l'avantage d'un attaquant ayant accès à un nombre polynomial de signatures est négligeable.

Le lemme suivant nous sera utile pour démontrer le Théorème 2.3. Il s'agit du Lemme 2.11 dans [Lyu08b] appliqué à nos paramètres.

Lemme 2.5. *Soient* $\mathbf{a}$ *un polynôme quelconque de l'ensemble* $D_{s,c}$ *et* $\mathbf{b}$ *un polynôme choisi de manière uniformément aléatoire dans* $D_{s,c}$*. Alors*

$$\mathsf{Pr}\left[\|\mathbf{ab} \bmod (x^n+1)\|_\infty \geq \sqrt{n}\log n\right] \leq 4ne^{\frac{-\log^2 n}{8}}.$$

Remarque 2.7. *Ce lemme sera utilisé pour montrer que pour deux clés secrètes, une fraction écrasante des polynômes de* $D_{s,c}$ *permet d'obtenir des polynômes de petite norme lorsqu'ils sont multipliés par l'une des clés. Il est important de remarquer que ce lemme est valable pour n'importe quel polynôme dans* $D_{s,c}$*, c'est-à-dire même si les polynômes sont choisis par l'attaquant.*

Théorème 2.3. *Pour* $b \in \{0,1\}$*, soit* $X_{b,\mathcal{P},sk_{i_b},\mu,R}$ *la variable aléatoire qui représente la sortie de* $\mathsf{Ring{-}sign}(\mathcal{P}, sk_{i_b}, \mu, R)$*, où* $\mathcal{P} = (k, n, m_u, p, S)$*,* sk_{i_b}, μ *et* R *sont des entrées arbitraires pour l'algorithme de signature. Si l'ensemble de ces paramètres est validé par l'étape 0 de l'algorithme de signature, alors nous avons que :*

$$\Delta(X_{0,\mathcal{P},sk_{i_0},\mu,R}, X_{1,\mathcal{P},sk_{i_1},\mu,R}) = n^{-\omega(1)}.$$

Démonstration. Pour avoir des notations plus simples, nous remplaçons les variables aléatoires $X_{0,\mathcal{P},sk_{i_0},\mu,R}$ et $X_{1,\mathcal{P},sk_{i_1},\mu,R}$ du théorème par X_0 et X_1. La signature

générée par l'algorithme de signature Ring−sign consiste en un vecteur de $\ell+1$ cordonnées. Nous notons la variable aléatoire qui correspond à la i-ème cordonnée de X_b par $X_b^{(i)}$ pour $b \in \{0,1\}$ et $i \in [\ell+1]$. Supposons qu'aucune de ces deux variables n'est égale à $\{\texttt{failed}\}$. Nous pouvons alors garantir que les paramètres donnés dans le théorème vérifient l'étape 0 de l'algorithme de signature Ring−sign.

Comme dans [Lyu08b], nous commençons par remarquer que l'ensemble

$$D_{s,c}(sk_{i_0}, sk_{i_1}) = \{\mathbf{c} \in D_{s,c} : \|sk_{i_0}\mathbf{c}\|_\infty, \|sk_{i_1}\mathbf{c}\|_\infty \leq \sqrt{n}\log n\}$$

a un cardinal très proche à celui de $D_{s,c}$.

Le Lemme 2.5 nous garantit que

$$\frac{|D_{s,c}(sk_{i_0}, sk_{i_1})|}{|D_{s,c}|} = 1 - n^{-\omega(1)}. \tag{2.4}$$

On note que n est un paramètre choisi par l'attaquant, mais le fait que n a passé l'étape 0 de l'algorithme de signature assure que $n \geq k$ et par conséquent, $n^{-\omega(1)}$ est une fonction négligeable. En divisant la distance statistique en deux, en fonction de si $\boldsymbol{\beta}$ appartient à $D_{s,c}(sk_{i_0}, sk_{i_1})$ ou pas, nous obtenons :

$$\begin{aligned}\Delta(X_0, X_1) &= \frac{1}{2}\sum_{\hat{\boldsymbol{\alpha}}_i \in D_z^{m_u}; i\in[\ell], \boldsymbol{\beta}\notin D_{s,c}(sk_{i_0}, sk_{i_1})} |\mathsf{Pr}[X_0 = (\hat{\boldsymbol{\alpha}}_i; i \in [\ell], \boldsymbol{\beta})] - \mathsf{Pr}[X_1 = (\hat{\boldsymbol{\alpha}}_i; i \in [\ell], \boldsymbol{\beta})]| \qquad (2.5)\\ &+ \frac{1}{2}\sum_{\hat{\boldsymbol{\alpha}}_i \in D_z^{m_u}; i\in[\ell], \boldsymbol{\beta}\in D_{s,c}(sk_{i_0}, sk_{i_1})} |\mathsf{Pr}[X_0 = (\hat{\boldsymbol{\alpha}}_i; i \in [\ell], \boldsymbol{\beta})] - \mathsf{Pr}[X_1 = (\hat{\boldsymbol{\alpha}}_i; i \in [\ell], \boldsymbol{\beta})]|. \qquad (2.6)\end{aligned}$$

Afin de prouver que la distance statistique est négligeable, nous allons tout d'abord montrer que (2.5) est négligeable et ensuite nous montrerons que (2.6) est égale à zéro. La dernière coordonnée de la signature est obtenue grâce à l'oracle aléatoire, donc la probabilité qu'elle ne soit pas dans l'ensemble $D_{s,c}(sk_{i_0}, sk_{i_1})$ est négligeable. En utilisant le fait que cela est vrai pour les deux variables X_0 et X_1 et que

$$\begin{aligned}\sum |\mathsf{Pr}[X_0 = (\hat{\boldsymbol{\alpha}}_i; i \in [\ell], \boldsymbol{\beta})] - \mathsf{Pr}[X_1 = (\hat{\boldsymbol{\alpha}}_i; i \in [\ell], \boldsymbol{\beta})]| &\leq \sum \mathsf{Pr}[X_0 = (\hat{\boldsymbol{\alpha}}_i; i \in [\ell], \boldsymbol{\beta})]\\ &+ \sum \mathsf{Pr}[X_1 = (\hat{\boldsymbol{\alpha}}_i; i \in [\ell], \boldsymbol{\beta})],\end{aligned}$$

nous pouvons montrer que :

$$(2.5) \leq 1 - \frac{|D_{s,c}(sk_{i_0}, sk_{i_1})|}{|D_{s,c}|} = n^{-\omega(1)}. \tag{2.7}$$

Plus formellement, cela donne :

$$(2.5) \leq \frac{1}{2}\sum_{\hat{\boldsymbol{\alpha}}_i \in D_z^{m_u}; i\in[\ell], \boldsymbol{\beta}\notin D_{s,c}(sk_{i_0}, sk_{i_1})} \mathsf{Pr}[X_0 = (\hat{\boldsymbol{\alpha}}_i; i \in [\ell], \boldsymbol{\beta})] + \mathsf{Pr}[X_1 = (\hat{\boldsymbol{\alpha}}_i; i \in [\ell], \boldsymbol{\beta})].$$

Pour $b \in \{0,1\}$, en notant que $\sum_{\forall A} \mathsf{Pr}[A \wedge B] = \sum_{\forall A} \mathsf{Pr}[A \mid B]\mathsf{Pr}[B] = \mathsf{Pr}[B]$, nous obtenons

$$\sum_{\hat{\boldsymbol{\alpha}}_i \in D_z^{m_u}; i\in[\ell], \boldsymbol{\beta}\notin D_{s,c}(sk_{i_0}, sk_{i_1})} \mathsf{Pr}[X_b = (\hat{\boldsymbol{\alpha}}_i; i \in [\ell], \boldsymbol{\beta})] = \sum_{\boldsymbol{\beta}\notin D_{s,c}(sk_{i_0}, sk_{i_1})} \mathsf{Pr}\left[X_b^{(\ell+1)} = \boldsymbol{\beta}\right].$$

Enfin, comme $X_b^{(l+1)}$ est obtenue grâce à un appel à l'oracle aléatoire $H(\sum_{i\in[\ell]} h_i(\hat{\mathbf{y}}_i), R, \mu)$, on a $\mathsf{Pr}\left[X_b^{(\ell+1)} = \boldsymbol{\beta}\right] = 1/|D_{s,c}|$. Il est important de remarquer que cela est vrai indépendamment de la distribution de l'entrée pour H dans le modèle de l'oracle aléatoire. Par conséquent, même si les h_i dans $H(\sum h_i(\hat{\mathbf{y}}_i), R, \mu)$ ont été choisies par un attaquant (par exemple, $h_i(\mathbf{x}) = 0$ pour tout $\mathbf{x}$ et i), la probabilité reste la même. En utilisant cette probabilité pour tout $\boldsymbol{\beta} \notin D_{s,c}(sk_{i_0}, sk_{i_1})$, nous obtenons immédiatement (2.7).

Afin de prouver que (2.6) est égale à zéro, nous montrons que chaque terme de la somme est nul. On sait que la dernière cordonnée de la signature est générée grâce à un appel à l'oracle aléatoire H, donc on a la même probabilité d'obtenir $\boldsymbol{\beta}$ pour les deux variables. Par conséquent, nous devons prouver que pour chaque terme dans (2.6) :

$$\mathsf{Pr}\left[(X_0^{(i)}; i \in [\ell]) = (\hat{\boldsymbol{\alpha}}_i; i \in [\ell]) | X_0^{(\ell+1)} = \boldsymbol{\beta}\right] = \mathsf{Pr}\left[(X_1^{(i)}; i \in [\ell]) = (\hat{\boldsymbol{\alpha}}_i; i \in [\ell]) | X_1^{(\ell+1)} = \boldsymbol{\beta}\right] \quad (2.8)$$

Nous allons montrer que pour $b \in \{0, 1\}$, $\mathsf{Pr}[X_b^{(i)} = \hat{\boldsymbol{\alpha}}_i | X_b^{(\ell+1)} = \boldsymbol{\beta}]$ est égale à $1/|D_z^{m_u}|$ si $i \in [\ell] \setminus i_b$ et est égale à $1/|D_y^{m_u}|$ si $i = i_b$. Cela suffit pour démontrer (2.8), puisque les ℓ premières coordonnées des variables aléatoires ont été choisies de manière indépendante dans l'algorithme de signature.

Nous notons par $\hat{\mathbf{y}}_{b,i}$ la variable $\hat{\mathbf{y}}_i$ qui correspond à l'exécution de l'algorithme de signature dans laquelle la clé secrète sk_{i_b} est utilisée. Pour $i \neq i_b$, nous avons $X_b^{(i)} = \hat{\boldsymbol{\alpha}}_i$ si $\hat{\mathbf{y}}_{b,i} = \hat{\boldsymbol{\alpha}}_i$. Puisque $\hat{\mathbf{y}}_{b,i}$ est choisie de manière uniformément aléatoire dans $D_z^{m_u}$, et $\hat{\boldsymbol{\alpha}}_i \in D_z^{m_u}$, alors la probabilité que les deux valeurs soient égales est $1/|D_z^{m_u}|$. D'autre part, pour $i = i_b$, nous avons $X_b^{(i_b)} = \hat{\boldsymbol{\alpha}}_{i_b}$ si $\hat{\mathbf{y}}_{b,i_b} = \hat{\boldsymbol{\alpha}}_{i_b} - sk_{i_b}\boldsymbol{\beta}$. Étant donné que la variable $\hat{\mathbf{y}}_{b,i}$ est choisie de manière uniformément aléatoire dans $D_y^{m_u}$, la probabilité qu'une variable soit égale à une telle valeur est $1/|D_y^{m_u}|$, si cette variable est dans $D_y^{m_u}$ et 0 si non. Par la définition de $D_{s,c}(sk_{i_0}, sk_{i_1})$, l'ensemble à lequel $\boldsymbol{\beta}$ appartient, nous avons $sk_{i_b}\boldsymbol{\beta} \leq \sqrt{n}\log n$ ce qui implique que $\hat{\boldsymbol{\alpha}}_{i_b} - sk_{i_b}\boldsymbol{\beta}$ appartient à $D_y^{m_u}$ et ceci termine la preuve. □

2.4.2 Inforgeabilité contre les attaques à sous-cercles choisis

Dans cette section, nous montrerons que si un attaquant est capable de casser l'inforgeabilité contre les attaques à sous-cercle choisis de notre schéma, alors il sera capable de casser l'inforgeabilité du schéma LSig09 de Lyubashevsky (voir la section 1.7.2.1).

2.4.2.1 Inforgeabilité de la signature LSig09 avec des nouveaux paramètres

Dans cette section nous démontrons l'inforgeabilité du schéma de signature LSig09 dans le cas où ce schéma est instancié avec les paramètres que nous avons

proposés dans la Table 2.1. En particulier, nous montrons que le schéma satisfait l'inforgeabilité existentielle contre les attaques à messages choisis avec une fonction de hachage ayant $m = n^c \cdot m_u$ colonnes.

Avant de prouver ceci, nous commençons par montrer que les deux résultats clés (Lemme 1.7 et le Théorème 1.5) utilisés dans la preuve d'inforgeabilité donnée par Lyubashvesky, restent vrais pour nos paramètres.

Lemme 2.6 (Lemme 1.7 adapté à nos paramètres)**.** *Soit h une fonction de hachage de $\mathcal{H}(\mathcal{D}, D_h, m)$. Pour tout secret $\hat{\mathbf{s}}_0$ choisi de manière aléatoire et uniforme dans l'ensemble $D_{s,c}^m$, la probabilité qu'il n'existe pas $\hat{\mathbf{s}}_1 \in D_{s,c}^m$ tel que, $\hat{\mathbf{s}}_0 \neq \hat{\mathbf{s}}_1$ et $h(\hat{\mathbf{s}}_0) = h(\hat{\mathbf{s}}_1)$ est négligeable en le paramètre de sécurité.*

Démonstration. En suivant le même raisonnement de la preuve 1.7.2.1 du Lemme 1.7, il suffit de montrer que pour les paramètres de la Table 2.1 nous avons que $p^n/3^{mn}$ est négligeable en le paramètre de sécurité. En effet, pour $p = \Theta(n^{4+c})$, $m = (3 + 2c/3)n^c \log n$ nous avons que $p^n/3^{mn} = 2^{-\Omega(n \log n)}$. □

Soit $\mathcal{P} = (k, n, m, p)$ les paramètres publics de la Table 2.1 pour le schéma de signature LSig09. Soit $X_{sk,pk,\mathcal{P},\mu}$ une variable aléatoire qui représente la signature d'un message μ suivant la clé secrète sk.

Soit $h \in \mathcal{H}(\mathcal{D}, D_h, m)$, le théorème suivant montre que deux signatures du même message μ générées suivant deux clés secrètes $\hat{\mathbf{s}}_0, \hat{\mathbf{s}}_1 \in D_{s,c}$, en utilisant l'algorithme S−sign du schéma LSig09, telles que $h(\hat{\mathbf{s}}_0) = h(\hat{\mathbf{s}}_1)$ sont indistinguables.

Théorème 2.4 (Théorème 1.5 adapté à nos paramètres)**.** *Pour $b \in \{0, 1\}$, soit $X_{sk_b,pk,\mathcal{P},\mu}$ la variable aléatoire qui représente le résultat de l'algorithme de signature* S−sign$(\mathcal{P}, sk_b, pk, \mu)$*, telle que $\mathcal{P} = (k, n, m, p)$, $\mu \in \{0, 1\}^\star$ et $(sk_b = \hat{\mathbf{s}}_b; b \in \{0, 1\}, pk = h)$. Nous avons*

$$\Delta(X_{\hat{\mathbf{s}}_0,h,\mathcal{P},\mu}, X_{\hat{\mathbf{s}}_1,h,\mathcal{P},\mu}) = n^{-\omega(1)}.$$

Démonstration. Pour démontrer le théorème, nous allons adapter la preuve donnée par Lyuvashevsky dans [Lyu08b] à nos paramètres.

Afin d'avoir des notations plus légères, nous notons les deux variables aléatoires du théorème par X_0 et X_1. Étant donné que la sortie de l'algorithme de signature est composée de deux éléments, nous notons par $X_b^{(i)}$ pour $i \in \{1, 2\}$ et $b \in \{0, 1\}$ la variable aléatoire qui correspond à la i-ème composante de X_b.

Comme dans [Lyu08b], nous commençons par définir l'ensemble suivant :

$$D_{s,c}(\hat{\mathbf{s}}_0, \hat{\mathbf{s}}_1) = \{\mathbf{c} \in D_{s,c} : \|\hat{\mathbf{s}}_0\mathbf{c}\|_\infty, \|\hat{\mathbf{s}}_1\mathbf{c}\|_\infty \leq \sqrt{n} \log n\} \tag{2.9}$$

La probabilité qu'un vecteur $\mathbf{c}$ choisit aléatoirement dans $D_{s,c}$ soit dans $D_{s,c}(\hat{\mathbf{s}}_0, \hat{\mathbf{s}}_1)$ est

$$\frac{\mid D_{s,c}(\hat{\mathbf{s}}_0, \hat{\mathbf{s}}_1) \mid}{\mid D_{s,c} \mid} = 1 - n^{-\omega(1)}.$$

Cette probabilité est dûe au Lemme 2.5. Nous pouvons remarquer que la distance statistique entre les deux variables aléatoires X_0 et X_1 peut s'écrire comme suit ;

$$\begin{aligned}\Delta(X_0, X_1) &= \frac{1}{2} \sum_{\hat{\boldsymbol{\alpha}} \in D_z^m, \boldsymbol{\beta} \notin D_{s,c}(\hat{\mathbf{s}}_0, \hat{\mathbf{s}}_1)} |\mathsf{Pr}[X_0 = (\hat{\boldsymbol{\alpha}}, \boldsymbol{\beta})] - \mathsf{Pr}[X_1 = (\hat{\boldsymbol{\alpha}}, \boldsymbol{\beta})]| \quad (2.10) \\ &+ \frac{1}{2} \sum_{\hat{\boldsymbol{\alpha}} \in D_z^m, \boldsymbol{\beta} \in D_{s,c}(\hat{\mathbf{s}}_0, \hat{\mathbf{s}}_1)} |\mathsf{Pr}[X_0 = (\hat{\boldsymbol{\alpha}}, \boldsymbol{\beta})] - \mathsf{Pr}[X_1 = (\hat{\boldsymbol{\alpha}}, \boldsymbol{\beta})]| \quad (2.11)\end{aligned}$$

Cette équation est pratiquement identique à celle obtenue dans la preuve 2.4.1.3 du Théorème 2.3 dans la section précédente et nous allons suivre le même principe de preuve. Nous allons tout d'abord montrer que la valeur de l'équation (2.10) est négligeable. Plus précisément, nous montrerons que

$$(2.10) \leq \mathsf{Pr}_{\mathbf{e} \leftarrow H}[\mathbf{e} \notin D_{s,c}] = 1 - n^{-\omega(1)}. \quad (2.12)$$

On peut écrire grâce à l'inégalité triangulaire

$$(2.10) \leq \frac{1}{2} \sum_{\hat{\boldsymbol{\alpha}} \in D_z^m, \boldsymbol{\beta} \notin D_{s,c}(\hat{\mathbf{s}}_0, \hat{\mathbf{s}}_1)} \mathsf{Pr}[X_0 = (\hat{\boldsymbol{\alpha}}, \boldsymbol{\beta})] + \mathsf{Pr}[X_1 = (\hat{\boldsymbol{\alpha}}, \boldsymbol{\beta})] \quad (2.13)$$

En utilisant le fait que $\sum_{\forall A} \mathsf{Pr}[A \wedge B] = \sum_{\forall A} \mathsf{Pr}[A \mid B]\mathsf{Pr}[B] = \mathsf{Pr}[B]$, nous obtenons que pour $b \in \{0, 1\}$:

$$\sum_{\hat{\boldsymbol{\alpha}} \in D_z^m, \boldsymbol{\beta} \notin D_{s,c}(\hat{\mathbf{s}}_0, \hat{\mathbf{s}}_1)} \mathsf{Pr}[X_b = (\hat{\boldsymbol{\alpha}}, \boldsymbol{\beta})] = \sum_{\boldsymbol{\beta} \notin D_{s,c}(\hat{\mathbf{s}}_0, \hat{\mathbf{s}}_1)} \sum_{\hat{\boldsymbol{\alpha}} \in D_z^m} \mathsf{Pr}[X_b^{(1)} = \hat{\boldsymbol{\alpha}} \mid X_b^{(2)} = \boldsymbol{\beta}]\mathsf{Pr}[X_b^{(2)} = \boldsymbol{\beta}].$$

Par conséquent pour $b \in \{0, 1\}$, nous avons :

$$\sum_{\hat{\boldsymbol{\alpha}} \in D_z^m, \boldsymbol{\beta} \notin D_{s,c}(\hat{\mathbf{s}}_0, \hat{\mathbf{s}}_1)} \mathsf{Pr}[X_b = (\hat{\boldsymbol{\alpha}}, \boldsymbol{\beta})] = \sum_{\boldsymbol{\beta} \notin D_{s,c}(\hat{\mathbf{s}}_0, \hat{\mathbf{s}}_1)} \mathsf{Pr}\left[X_b^{(2)} = \boldsymbol{\beta}\right] = 1 - \sum_{\boldsymbol{\beta} \in D_{s,c}(\hat{\mathbf{s}}_0, \hat{\mathbf{s}}_1)} \mathsf{Pr}\left[X_b^{(2)} = \boldsymbol{\beta}\right].$$

Nous avons que $X_b^{(2)}$ est obtenu grâce à un appel à l'oracle aléatoire H, qui a une sortie uniforme, par conséquent $\mathsf{Pr}\left[X_b^{(2)} = \boldsymbol{\beta}\right] = 1/|D_{s,c}|$ et nous obtenons que :

$$\sum_{\boldsymbol{\beta} \in D_{s,c}(\hat{\mathbf{s}}_0, \hat{\mathbf{s}}_1)} \mathsf{Pr}\left[X_b^{(2)} = \boldsymbol{\beta}\right] = |D_{s,c}(\hat{\mathbf{s}}_0, \hat{\mathbf{s}}_1)| \cdot \frac{1}{|D_{s,c}|} = 1 - n^{-\omega(1)}.$$

Pour finir la preuve nous montrerons, encore, que l'équation (2.11) est égale à zéro, en montrant que chaque terme de la somme dans (2.11) est nul. Nous commençons par remarquer que pour $b \in \{0, 1\}$:

$$\begin{aligned}\mathsf{Pr}[X_b = (\hat{\boldsymbol{\alpha}}, \boldsymbol{\beta})] &= \mathsf{Pr}\left[(X_b^{(1)} = \hat{\boldsymbol{\alpha}}) \wedge (X_b^{(2)} = \boldsymbol{\beta})\right] \\ &= \mathsf{Pr}\left[(X_b^{(1)} = \hat{\boldsymbol{\alpha}}) \mid (X_b^{(2)} = \boldsymbol{\beta})\right] \cdot \mathsf{Pr}\left[X_b^{(2)} = \boldsymbol{\beta}\right]\end{aligned}$$

Nous savons que le deuxième terme de chaque signature est obtenu après un appel à l'oracle aléatoire H, donc nous avons $\mathsf{Pr}\left[X_1^{(2)} = \boldsymbol{\beta}\right] = \mathsf{Pr}\left[X_2^{(2)} = \boldsymbol{\beta}\right]$. Par conséquent, il suffit de montrer que

$$\mathsf{Pr}\left[(X_0^{(1)} = \hat{\boldsymbol{\alpha}}) \mid (X_0^{(2)} = \boldsymbol{\beta})\right] = \mathsf{Pr}\left[(X_1^{(1)} = \hat{\boldsymbol{\alpha}}) \mid (X_1^{(2)} = \boldsymbol{\beta})\right] \tag{2.14}$$

Nous allons montrer que pour $b \in \{0,1\}$, $\mathsf{Pr}\left[(X_b^{(1)} = \hat{\boldsymbol{\alpha}}) \mid (X_b^{(2)} = \boldsymbol{\beta})\right]$ est égale à $1/|D_y^m|$. Pour $b \in \{0,1\}$, nous avons $X_b^{(1)} = \hat{\boldsymbol{\alpha}}_b$ si $\hat{\mathbf{y}}_b = \hat{\boldsymbol{\alpha}}_b - sk_b\boldsymbol{\beta}$. Puisque la variable $\hat{\mathbf{y}}_b$ est choisie de manière uniformément aléatoire dans D_y^m, la probabilité qu'une variable aléatoire soit égale à une telle valeur est $1/|D_y^m|$ si cette variable est dans D_y^m et 0 si non. Par la définition de l'ensemble $D_{s,c}(\hat{\mathbf{s}}_0, \hat{\mathbf{s}}_1)$, l'ensemble auquel $\boldsymbol{\beta}$ appartient, nous avons que $sk_b\boldsymbol{\beta} \leq \sqrt{n}\log n$ par conséquent $\hat{\boldsymbol{\alpha}}_b - sk_b\boldsymbol{\beta} \in D_y^m$. □

Corollaire 2.2. *S'il existe un algorithme en temps polynomial qui peut casser l'inforgeabilité existentielle du schéma* LSig09 *pour les paramètres présentés dans la Table 2.1, et plus précisément pour des fonctions de hachage ayant $m = n^c \cdot m_u$ colonnes. Alors il existe un algorithme polynomial qui peut résoudre (x^n+1)-SVP$_\gamma^\infty$ dans le pire cas pour $\gamma = O(n^{2,5+2c})$.*

Démonstration. La preuve d'inforgeabilité du schéma de signature LSig09 (voir la section 1.7.2.1 pour plus de détails) est divisée en deux parties. La première partie (Théorème 1.6) montre que si on peut forger une signature alors il est possible de trouver des collisions sur les fonctions de hachage dans $\mathcal{H}(\mathcal{D}, D_h, m)$, avec $\mathcal{D}, D_h$ et m sont les paramètres de la Table 1.2. La deuxième partie montre que trouver des collisions sur ces fonctions de hachage est aussi difficile que résoudre le problème (x^n+1)-SVP$_\gamma^\infty$ dans le pire cas pour un certain γ.

Pour nos paramètres, le Théorème 1.6 est presque inchangé. Comme nous avons modifié le paramètre m qui est le nombre de colonnes des fonctions de hachage, le seul point qui pourrait poser problème est lorsque le Lemme 1.7 et le Théorème 1.5 sont appliqués. Ceux-ci montrent que pour les paramètres du schéma LSig09 il existe avec une bonne probabilité une deuxième pré-image pour un certain haché et que les signatures issues suivant ces deux pré-images sont indistinguables. Le Lemme 2.6 et le Théorème 2.4 que nous venons de prouver montrent qu'avec des fonctions de hachage ayant m colonnes, ces propriétés restent vraies.

Par conséquent, en utilisant le Théorème 1.6 nous déduisons que si un attaquant est capable de forger des signatures avec nos paramètres (paramètres de la Table 2.1) alors, il peut trouver des collisions sur des fonctions de hachage dans $\mathcal{H}(\mathcal{D}, D_h, m)$. Finalement, en utilisant le Théorème 2.1 avec nos paramètres nous obtenons que trouver des collisions est aussi difficile que résoudre (x^n+1)-SVP$_\gamma^\infty$ dans le pire cas pour $\gamma = O(n^{2,5+2c})$ (au lieu de $O(n^{2,5})$ pour le schéma LSig09). □

2.4.2.2 Preuve d'inforgeabilité contre les attaques à sous-cercle choisis

Dans la preuve d'inforgeabilité du schéma de signature LSig09 (voir la section 1.7.2.1 pour plus de détails), il est montré que si un attaquant est capable de casser l'inforgeabilité de ce schéma alors on peut utiliser cet attaquant pour trouver une collision sur une fonction de hachage définie par un vecteur de m' polynômes donné comme challenge au début du jeu. Dans notre signature de cercle, le challenge du jeu d'inforgeabilité est un ensemble de fonctions de hachage défini par un ensemble de ℓ vecteurs de m_u polynômes. L'idée est donc de définir $m' = \ell \cdot m_u$, puis transformer ce challenge en ℓ vecteurs et montrer que si nous donnons ça comme challenge à l'attaquant capable de forger des signatures pour notre schéma, alors nous pouvons obtenir une signature forgée et valide pour le schéma LSig09 avec une probabilité non-négligeable.

Il y a deux problèmes concernant les clés. Le premier problème vient de la distribution des clés publiques. Dans le schéma LSig09 les polynômes du m'-uplet sont générés de manière uniformément aléatoire. En revanche les polynômes dans les m_u-uplets que l'attaquant s'attend à recevoir ne le sont pas. Ceci est dû au fait que dans l'algorithme de génération de clés $\mathsf{Ring-gen-keys}$, il y a toujours un polynôme dans le m_u-uplet qui est calculé à partir des autres $m_u - 1$ polynômes (voir l'étape (5) de $\mathsf{Ring-gen-keys}$). Ceci n'est pas exploitable par un attaquant car nous avons montré dans le Corollaire 2.1, que la distance statistique entre les clés publiques de notre schéma de signature de cercle et des fonctions de hachage choisies de manière uniformément aléatoire est négligeable. Ceci implique que si un attaquant est capable de forger des signatures avec des fonctions de hachage associées aux clés publiques, alors il est aussi capable de les forger avec des fonctions de hachage choisies de manière uniformément aléatoire. En effet, si on modélise l'attaquant par une fonction randomisée, alors il ne peut pas augmenter la distance statistique entre les deux familles des fonctions de hachage. Par conséquent, s'il réussi à mener à bien son attaque avec une probabilité non négligeable avec une famille alors il réussira avec l'autre aussi.

Le deuxième problème est que nous ne connaissons pas les clés secrètes associées aux challenges donnés à l'attaquant et nous sommes obligés de répondre à ses requêtes de signature. Nous résolvons ce problème en programmant l'oracle aléatoire utilisé par l'attaquant.

Nous allons maintenant montrer que notre schéma de signature de cercle présenté dans la section 2.3 vérifie l'inforgeabilité contre les attaques à sous-cercles choisis.

Théorème 2.5. *S'il existe un algorithme en temps polynomial qui peut gagner le jeu d'inforgeabilité contre les attaques à sous-cercles choisis pour notre schéma de signature de cercle, alors il existe un algorithme en temps polynomial qui peut résoudre* $(x^n + 1)$-$\mathsf{SVP}_\gamma^\infty$ *dans le pire cas pour* $\gamma = \tilde{O}(n^{2.5+2c})$.

Démonstration. Supposons qu'on a un adversaire $\mathcal{A}$ qui peut avec une probabilité non négligeable forger une signature de notre schéma de signature de cercle. Étant donné le Corollaire 2.2, il suffit de montrer qu'en utilisant $\mathcal{A}$, nous pouvons concevoir un algorithme de temps polynomial $\mathcal{B}$ qui peut forger une signature pour le schéma LSig09 avec un avantage non-négligeable pour les paramètres donnés dans la Table 2.1.

Dans la suite nous montrons comment construire $\mathcal{B}$ et nous montrons comment $\mathcal{B}$ utilise l'adversaire $\mathcal{A}$ pour forger des signatures du schéma LSig09.

Préparation : $\mathcal{B}$ reçoit la description de la fonction de hachage (c'est-à-dire un vecteur de $m' = \ell \cdot m_u$ polynômes $(\mathbf{a}_1, \ldots, \mathbf{a}_{\ell \cdot m_u}) \in \mathcal{D}$) et $\mathbf{S} \in \mathcal{D}$ comme challenge. Il a accès à l'oracle aléatoire H_L de l'algorithme de signature. L'adversaire $\mathcal{B}$ divise l'ensemble des polynômes en ℓ ensembles de m_u polynômes $(\mathbf{a}_{i,1}, \ldots, \mathbf{a}_{i,m_u})$ pour $i \in [\ell]$. Enfin, $\mathcal{B}$ initialise $\mathcal{A}$ en lui donnant les ℓ vecteurs de m_u polynômes, les paramètres publics associés (parmi lesquels $\mathbf{S}$) et l'accès à l'oracle aléatoire de l'algorithme de signature de cercle H contrôlé par lui même.

Interaction avec les oracles : $\mathcal{B}$ répond aux requêtes de $\mathcal{A}$ pour l'oracle aléatoire et l'oracle de signature comme suit. Pour chaque requête (x_y, x_h, x_m) à l'oracle aléatoire, il teste s'il a déjà répondu à une telle requête. Si c'est le cas, il répond toujours avec le même résultat. Dans le cas inverse, il répond par $H_L(x_y, x_h \| x_m)$, et il enregistre le résultat dans une liste. Pour chaque requête de signature $(\{h_i\}_{i \in T}, i_0, \mu)$ telle que $i_0 \in T \subseteq [\ell]$, $\mathcal{B}$ programme l'oracle aléatoire H pour produire une signature valide. En d'autres termes :

1. Il suit les étapes du schéma de signature en générant un ensemble de $\hat{\mathbf{y}}_i \leftarrow D_z^{m_u}$ pour $i \in T$.
2. Il génère de manière uniformément aléatoire $\mathbf{r} \leftarrow D_{s,c}$.
3. Il vérifie si H_L a été appelé avec des paramètres $(\sum_i h_i(\hat{\mathbf{y}}_i) - \mathbf{Sr}, \{h_i\}_{i \in T} \| \mu)$. Si c'est le cas il abandonne le jeu.
4. Il programme l'oracle aléatoire H de sorte que $H(\sum_i h_i(\hat{\mathbf{y}}_i) - \mathbf{Sr}, \{h_i\}_{i \in T}, \mu) = \mathbf{r}$, et il enregistre le résultat dans une liste.
5. Finalement, il retourne $(\hat{\mathbf{y}}_i; i \in T, \mathbf{r})$ comme signature.

Génération de la signature : À un moment donné, l'attaquant $\mathcal{A}$ termine son exécution et retourne avec une probabilité non-négligeable une signature $((\hat{\mathbf{z}}_i; i \in T, \mathbf{e}), \mu, \{h_i\}_{i \in T})$, telle que $T \subseteq [\ell]$. $\mathcal{B}$ fait un padding des coordonnées restantes de la signature avec des polynômes nuls $\hat{\mathbf{z}}_i = 0$ pour $i \in [\ell] \backslash T$, ensuite il retourne $((\hat{\mathbf{z}}_1 \| \cdots \| \hat{\mathbf{z}}_\ell, \mathbf{e}), \mu)$, tel que $\hat{\mathbf{z}}_1 \| \cdots \| \hat{\mathbf{z}}_\ell$, est le vecteur des polynômes obtenus après la concaténation de chacun des vecteurs de polynômes $\hat{\mathbf{z}}_i$.

Analyse : Nous allons détailler complètement le raisonnement de la preuve.

Tout d'abord, nous remarquons que dans l'étape d'interaction avec les oracles il y a une possibilité d'abandon mais d'après le Lemme 2.7 ci-après cet événement est négligeable.

Afin de prouver que $\mathcal{A}$ pourrait forger une signature de cercle avec une probabilité non négligeable, nous devons montrer que toutes les entrées qu'il reçoit ont une distribution proche de celle qu'il aurait reçu dans un challenge d'une signature de cercle. Une fois que cela est démontré, nous devons aussi prouver que la signature forgée par $\mathcal{B}$ est valide pour le schéma LSig09.

Notons d'abord que par le Corollaire 2.1, nous avons que la distribution du challenge donné au début du jeu à $\mathcal{A}$ est proche du point de vue de la distance statistique de celle des challenges de la signature de cercle. Une autre partie des entrées à considérer c'est les signatures de cercle générées par $\mathcal{B}$. D'après le Lemme 2.8 ci-après nous avons que la distribution des signatures générées par $\mathcal{B}$ est statistiquement proche de celles obtenues à partir de notre schéma de signature.

Nous pouvons donc assurer que $\mathcal{A}$ peut forger une signature de cercle avec une probabilité non négligeable. Maintenant, il reste à prouver que la signature forgée par $\mathcal{B}$ est valide. L'idée de base que nous utiliserons est que dans $((\hat{\mathbf{z}}_i; i \in T, \mathbf{e}), \mu, \{h_i\}_{i\in T})$, (i.e la signature forgée par $\mathcal{A}$) si $\mathbf{e}$ a été obtenu grâce à un appel direct à l'oracle aléatoire alors nous pouvons prouver que la signature forgée est valide pour le schéma LSig09. Sinon, nous utiliserons les mêmes idées de la preuve du Théorème 1.6 (voir l'annexe B) pour montrer que la probabilité que $\mathcal{A}$ devine $\mathbf{e}$ sans envoyer de requêtes à H est négligeable.

Nous remarquons que si $\mathcal{A}$ n'a pas envoyé la requête $(\sum_i h_i(\hat{\mathbf{z}}_i) - \mathbf{Se}, \{h_i\}_{i\in T}, \mu)$ à H, alors la probabilité que $\mathcal{A}$ devine $\mathbf{e}$ tel que $\mathbf{e} = H((\sum_i h_i(\hat{\mathbf{z}}_i) - \mathbf{Se}, \{h_i\}_{i\in T}, \mu))$ est $1/|D_{s,c}|$ ce qui est négligeable. Supposons que $\mathbf{e}$ a été obtenu grâce à un appel à l'oracle aléatoire H, alors il existe deux possibilités : ou bien $\mathbf{e}$ a été généré durant une requête de signature, ou il a été généré par une requête directe à l'oracle aléatoire. Nous laissons le deuxième cas pour la fin de la preuve.

Supposons que $\mathbf{e}$ a été généré durant une requête de signature qui a donnée comme résultat $(\hat{\mathbf{z}}'_i; i \in T', \mathbf{e})$ pour un cercle $\{h_i\}_{i\in T'}$ et un message μ'. Pour être valide, la signature forgée doit être différente de $((\hat{\mathbf{z}}'_i; i \in T', \mathbf{e}), \mu', \{h_i\}_{i\in T'})$ et par conséquent nous devons avoir soit $\{h_i\}_{i\in T'}\|\mu' \neq \{h_i\}_{i\in T}\|\mu$ ou bien $(\hat{\mathbf{z}}'_i; i \in T') \neq (\hat{\mathbf{z}}_i; i \in T)$. Le premier cas ne peut se produire qu'avec une probabilité négligeable car cela implique une collision sur H. En effet, $\{h_i\}_{i\in T'}\|\mu' \neq \{h_i\}_{i\in T}\|\mu$ implique que $\{h_i\}_{i\in T'} \neq \{h_i\}_{i\in T}$ ou bien $\mu' \neq \mu$ et dans les deux cas nous avons une collision puisque $H(\sum_{i\in T'} h_i(\hat{\mathbf{z}}'_i) - \mathbf{Se}, \{h_i\}_{i\in T'}, \mu') = H(\sum_{i\in T} h_i(\hat{\mathbf{z}}_i) - \mathbf{Se}, \{h_i\}_{i\in T}, \mu) = \mathbf{e}$ (nous avons une seule image par H de deux pré-images qui sont différentes soit dans la deuxième ou dans la troisième coordonnée). Pour le deuxième cas (i.e. quand $(\hat{\mathbf{z}}'_i; i \in T') \neq (\hat{\mathbf{z}}_i; i \in T)$), en utilisant le même raisonnement, on a

$$\sum_{i\in T'} h_i(\hat{\mathbf{z}}'_i) - \mathbf{Se} = \sum_{i\in T} h_i(\hat{\mathbf{z}}_i) - \mathbf{Se} \quad \text{pour} \quad (\hat{\mathbf{z}}'_i; i \in T') \neq (\hat{\mathbf{z}}_i; i \in T),$$

avec une probabilité proche de un, car sinon on a une collision sur H. On étend $\hat{\mathbf{z}}$ et $\hat{\mathbf{z}}'$ de T et T' à $[\ell]$ en fixant $\hat{\mathbf{z}}_i = 0$ pour $i \in [\ell]\backslash T$ et $\hat{\mathbf{z}}'_i = 0$ pour $i \in [\ell]\backslash T'$, ce qui nous permet d'obtenir une collision sur une fonction de hachage aléatoire de $\mathcal{H}(\mathcal{D}, D_h, m')$. Si cet événement se produit avec une probabilité non négligeable, alors en utilisant le Théorème 2.1, nous déduisons que nous pouvons résoudre (x^n+1)-SVP$^{\infty}_{\gamma}$ dans le pire cas pour $\gamma = \tilde{O}(n^{2.5+2c})$.

Plaçons nous maintenant dans le cas où $\mathbf{e}$ a été généré par une requête directe à l'oracle aléatoire. Nous avons $((\hat{\mathbf{z}}_i; i \in T, \mathbf{e}), \mu, \{h_i\}_{i\in T})$ comme signature forgée et puisque la signature de cercle doit être valide nous avons que $H(\sum_{i\in T} h_i(\hat{\mathbf{z}}_i) - \mathbf{Se}, \{h_i\}_{i\in T}, \mu) = \mathbf{e}$. En notant $x_y = \sum_{i\in T} h_i(\hat{\mathbf{z}}_i) - \mathbf{Se}$ et compte tenu de l'algorithme utilisé par $\mathcal{B}$, nous avons aussi $H_L(x_y, \{h_i\}_{i\in T}\|\mu) = \mathbf{e}$. Si nous faisons un padding avec des $\hat{\mathbf{z}}_i$ tels que $\hat{\mathbf{z}}_i = 0$ pour $i \in [\ell] \setminus T$, alors nous avons $\sum_{i\in[\ell]} h_i(\hat{\mathbf{z}}_i) - \mathbf{Se} = x_y$ et par conséquent $H_L(\sum_{i\in[\ell]} h_i(\hat{\mathbf{z}}_i) - \mathbf{Se}, \{h_i\}_{i\in T}\|\mu) = \mathbf{e}$. Donc, nous avons $\sigma' = (\hat{\mathbf{z}}_1\| \cdots \|\hat{\mathbf{z}}_\ell, \mathbf{e})$ est une signature de $\mu' = \{h_i\}_{i\in T}\|\mu$ et comme nous n'avons pas fait de requête de signature dans le schéma de signature de base, alors (σ', μ') est une signature forgée valide.

□

Lemme 2.7. *La probabilité d'abandon dans l'étape (3) de l'algorithme de génération de signature de la preuve du Théorème 2.5 est négligeable.*

Démonstration. Nous allons montrer que la distance statistique entre $\sum_i h_i(\hat{\mathbf{y}}_i) - \mathbf{Sr}$ et la distribution uniforme sur $\mathcal{D}$ est négligeable. Par conséquent, la probabilité que $\mathcal{A}$ génère une requête pour l'oracle aléatoire et une requête de signature est négligeable.

Soient X_i la variable aléatoire suivant la distribution de $h_i(\hat{\mathbf{y}}_i)$ et U_i une variable aléatoire suivant la distribution uniforme sur $\mathcal{D}$. En utilisant le Lemme 1.1 (le *leftover hash lemma*), les tailles de D_y, p^n et m_u sont suffisamment grandes par rapport à la taille de $\mathcal{D}$ pour que la distance statistique entre la distribution de $h_i(\hat{\mathbf{y}}_i)$ et la distribution uniforme sur $\mathcal{D}$ soit négligeable pour tout $i \in [\ell]$.

Pour $i \in [\ell]$, soient X_i une variable aléatoire suivant la distribution de $h_i(\hat{\mathbf{y}}_i)$ et U_i une variable aléatoire uniforme dans $\mathcal{D}$. D'après la Proposition 1.2, nous avons que

$$\Delta((X_1, \cdots, X_\ell), (U_1, \cdots, U_\ell)) \leq \sum_i \Delta(X_i, U_i).$$

Soit F la fonction qui fait la somme des variables X_i puis soustrait $\mathbf{Sr}$ de la somme obtenue. En utilisant le fait qu'une fonction ne peut pas augmenter la distance statistique (voir Proposition 1.3), nous obtenons que

$$\Delta(F(X_1, \cdots, X_\ell), F(U_1, \cdots, U_\ell)) \leq \sum_i \Delta(X_i, U_i).$$

Pour finir la preuve, nous devons juste de montrer que $F(U_1, \cdots, U_\ell)$ suit une distribution uniforme sur $\mathcal{D}$. En effet, cela vient du fait que soustraire un élément

Hybride 0 :

1. Pour tout $i \in [\ell]$; $i \neq j$;$\hat{\mathbf{y}}_i \leftarrow D_z^{m_u}$
2. Pour $i = j$; $\hat{\mathbf{y}}_j \leftarrow D_y^{m_u}$
3. Soit $\mathbf{r} \leftarrow H(\sum_{i\in[\ell]} h_i(\hat{\mathbf{y}}_i), R, \mu)$
4. Pour $i = j$, $\hat{\mathbf{z}}_j \leftarrow \hat{\mathbf{s}}_j\mathbf{r} + \hat{\mathbf{y}}_j$
5. Si $\hat{\mathbf{z}}_j \notin D_z^{m_u}$, revenir à l'étape (2).
6. Pour $i \neq j$, $\hat{\mathbf{z}}_i = \hat{\mathbf{y}}_i$
7. Retourner la signature $\sigma = (\hat{\mathbf{z}}_i; i \in [\ell], \mathbf{r})$

FIGURE 2.1 – **L'algorithme hybride 0 de** Ring−sign**.**

Hybride 1 :

1. Pour tout $i \in [\ell]$; $i \neq j$;$\hat{\mathbf{y}}_i \leftarrow D_z^{m_u}$
2. Pour $i = j$; $\hat{\mathbf{y}}_j \leftarrow D_y^{m_u}$
3. Choisir $\mathbf{r} \leftarrow D_{s,c}$
4. Pour $i = j$, $\hat{\mathbf{z}}_j \leftarrow \hat{\mathbf{s}}_j\mathbf{r} + \hat{\mathbf{y}}_j$
5. Si $\hat{\mathbf{z}}_j \notin D_z^{m_u}$, revenir à l'étape (2).
6. Pour $i \neq j$, $\hat{\mathbf{z}}_i = \hat{\mathbf{y}}_i$
7. Retourner la signature $\sigma = (\hat{\mathbf{z}}_i; i \in [\ell], \mathbf{r})$
8. Programmer $H(\sum_{i\in[\ell]} h_i(\hat{\mathbf{z}}_i) - \mathbf{Sr}, R, \mu) = \mathbf{r}$

FIGURE 2.2 – **L'algorithme hybride 1 de** Ring−sign**.**

(dans notre cas $\mathbf{Sr}$) sur l'ensemble des éléments de $\mathcal{D}$ revient juste à permuter les éléments de l'anneau et par conséquent la distribution uniforme reste inchangée. Ceci fini la preuve puisque $\sum_i \Delta(X_i, U_i)$ est négligeable. □

Lemme 2.8. *La distribution des signatures générées par $\mathcal{B}$ est statistiquement proche de celles obtenues à partir de notre schéma de signature.*

Démonstration. Pour démontrer le lemme, nous allons prouver que les schémas Hybride 0,1,2 (voir Figures 2.1, 2.2, 2.3) sont indistinguables. Il est clair que Hybride 0 correspond à notre schéma de base (Ring−sign). L'algorithme utilisé par l'attaquant $\mathcal{B}$ dans le jeu d'inforgeabilité contre les attaques à sous-cercles choisis correspond à celui de Hybride 2.

Tout d'abord, nous montrons que Hybride 0 et Hybride 1 sont indistinguables. La différence entre Hybride 0 et 1 est que l'étape (3) de Hybride 0 est coupée en deux étapes (3) et (8) dans Hybride 1. Nous remarquons qu'un attaquant ne peut distinguer la sortie de Hybride 0 et 1 que dans le cas où on choisit (dans Hybride 1)

Hybride 2 :

1. Pour tout $i \in [\ell]$;$\hat{\mathbf{y}}_i \leftarrow D_z^{m_u}$
2. Choisir $\mathbf{r} \leftarrow D_{s,c}$
3. Pour tout $i \in [\ell]$; $\hat{\mathbf{z}}_i = \hat{\mathbf{y}}_i$
4. Retourner la signature $\sigma = (\hat{\mathbf{z}}_i; i \in [\ell], \mathbf{r})$
5. Programmer $H(\sum_{i\in[\ell]} h_i(\hat{\mathbf{z}}_i) - \mathbf{Sr}, R, \mu) = \mathbf{r}$

FIGURE 2.3 – **L'algorithme hybride 2 de** Ring−sign**.**

des $\hat{\mathbf{y}}_i$ pour $i \in [\ell]$ tels que $(\sum_i h_i(\hat{\mathbf{y}}_i), R, \mu)$ correspond à une requête qu'il aurait déjà fait à l'oracle aléatoire. Encore une fois, en utilisant le *leftover hash lemma* (Lemme 1.1) nous obtenons que la probabilité de cet événement est négligeable.

Puisque l'étape (8) de Hybride 1 et l'étape (5) de Hybride 2 sont les mêmes il nous suffira de montrer que la distribution de la sortie de Hybride 1 avant la huitième étape est proche de celle de Hybride 2 avant la cinquième étape. Nous montrons que la distance statistique entre les deux distributions est négligeable. Pour ceci nous nous servons des techniques et notations utilisées dans la preuve du Théorème 2.3.

Soient X_1 et X_2 deux variables aléatoires représentant les distributions que nous souhaitons évaluer pour Hybride 1 et Hybride 2. Comme dans la preuve du Théorème 2.3, nous divisons la distance statistique entre ces deux variables en deux. Nous avons donc

$$\Delta(X_1, X_2) = \frac{1}{2} \sum_{\hat{\boldsymbol{\alpha}}_i \in D_z^{m_u}; i\in[\ell], \boldsymbol{\beta} \notin D_{s,c}(sk_{i_0}, sk_{i_1})} |\mathsf{Pr}[X_1 = (\hat{\boldsymbol{\alpha}}_i; i \in [\ell], \boldsymbol{\beta})] - \mathsf{Pr}[X_2 = (\hat{\boldsymbol{\alpha}}_i; i \in [\ell], \boldsymbol{\beta})]| \quad (2.15)$$

$$+ \frac{1}{2} \sum_{\hat{\boldsymbol{\alpha}}_i \in D_z^{m_u}; i\in[\ell], \boldsymbol{\beta} \in D_{s,c}(sk_{i_0}, sk_{i_1})} |\mathsf{Pr}[X_1 = (\hat{\boldsymbol{\alpha}}_i; i \in [\ell], \boldsymbol{\beta})] - \mathsf{Pr}[X_2 = (\hat{\boldsymbol{\alpha}}_i; i \in [\ell], \boldsymbol{\beta})]|. \quad (2.16)$$

D'après la preuve du Théorème 2.3 nous avons

$$(2.15) \leq \frac{1}{2} \sum_{\boldsymbol{\beta} \notin D_{s,c}(sk_{i_0}, sk_{i_1})} \mathsf{Pr}\left[X_1^{(\ell+1)} = \boldsymbol{\beta}\right] + \mathsf{Pr}\left[X_2^{(\ell+1)} = \boldsymbol{\beta}\right] = n^{-\omega(1)}.$$

Nous allons maintenant montrer que la différence entre les deux probabilités de l'équation (2.16) est négligeable. Commençons par X_2.

$$\mathsf{Pr}[(X_2^{(i)}; i \in [\ell], X_2^{(\ell+1)}) = (\hat{\boldsymbol{\alpha}}_i; i \in [\ell], \boldsymbol{\beta})] = (1/|D_z^{m_u}|)^{\ell} \cdot (1/|D_{s,c}|).$$

D'autre part pour X_1 nous avons.

$$\begin{aligned}\mathsf{Pr}[(X_1^{(i)}; i \in [\ell], X_1^{(\ell+1)}) = (\hat{\boldsymbol{\alpha}}_i; i \in [\ell], \boldsymbol{\beta})] &= \mathsf{Pr}[(X_1^{(i)}; i \in [\ell]) = (\hat{\boldsymbol{\alpha}}_i; i \in [\ell]) | X_1^{(\ell+1)} = \boldsymbol{\beta}] \cdot \mathsf{Pr}[X_1^{(\ell+1)} = \boldsymbol{\beta}] \\ &= (1/|D_z^{m_u}|)^{\ell-1} \cdot (1/|D_y^{m_u}|) \cdot (1/|D_{s,c}|).\end{aligned}$$

En effet, nous avons que $\mathsf{Pr}[X^i = \hat{\boldsymbol{\alpha}}_i | X^{\ell+1} = \boldsymbol{\beta}]$ est égale à $1/|D_z^{m_u}|$ pour $i \in [\ell] \setminus j$

et elle est égale à $1/|D_y^{m_u}|$ si $i = j$. Nous avons que

$$(2.16) = \frac{1}{2} \cdot |D_{s,c}(sk_{i_0}, sk_{i_1})| \cdot \frac{1}{|D_{s,c}|} \cdot \left(\frac{1}{|D_z^{m_u}|}\right)^{\ell} \cdot \left(1 - \frac{|D_z^{m_u}|}{|D_y^{m_u}|}\right).$$

D'après l'équation (2.4), nous avons que

$$(2.16) \leq \left(1 - n^{-\omega(1)}\right) \cdot \left(1 - \frac{|D_z^{m_u}|}{|D_y^{m_u}|}\right) \cdot \left(\frac{1}{|D_z^{m_u}|}\right)^{\ell}.$$

D'après les paramètres de la Table 2.1, l'équation (2.16) est majorée par une quantité négligeable en n.

□

2.5 Schéma sûr contre les attaques de corruption interne

Supposons que nous avons un algorithme $\mathcal{A}$ capable de casser l'inforgeabilité contre corruption interne. Nous aimerions montrer qu'il existe un algorithme $\mathcal{B}$ qui peut casser l'inforgeabilité du schéma de signature LSig09 avec nos paramètres en utilisant $\mathcal{A}$. Le problème principal est que lorsque $\mathcal{B}$ obtient le challenge pour le schéma de signature de base, il peut le diviser en des vecteurs de polynômes mais il ne peut pas connaitre les clés secrètes correspondantes. Par conséquent, $\mathcal{B}$ ne peut pas répondre aux requêtes de corruption des clés secrètes.

Une idée serait de donner à $\mathcal{A}$ une version modifiée des vecteurs de polynômes de telle sorte que $\mathcal{B}$ connaîtrait les clés secrètes associées à ces vecteurs. Le problème avec cette idée est que nous n'avons pas réussi à trouver une façon pour exploiter la signature obtenue par $\mathcal{A}$. Une approche plus standard pour résoudre ce problème est celle utilisée par Wang et Sun dans [WS11], qui consiste à donner à $\mathcal{A}$ plus de vecteurs de polynômes que ceux obtenus par la division du challenge initiale. Ces vecteurs supplémentaires sont générés par l'algorithme de génération des clés et $\mathcal{B}$ connait les clés secrètes associées. S'il y en a assez de ces vecteurs, alors il est possible d'obtenir un cas dans lequel toutes les requêtes de corruption correspondent avec une probabilité non négligeable à des vecteurs dont $\mathcal{B}$ connait les clés secrètes.

Malheureusement, cette approche crée de nouveau problème pour le schéma Wang et Sun [WS11] ainsi que pour notre schéma. Par exemple, si la moitié des vecteurs ne correspondent pas au challenge, alors la réponse final de l'attaquant va correspondre à un cercle qui contient des vecteurs qui ne font pas partie de notre challenge avec une probabilité proche de un. Dans certains schémas dont la sécurité repose sur des problèmes difficiles de la théorie de nombre, ceci n'est pas un problème, mais dans notre cas telles réponses (signatures forgées) sont inutiles

puisque $\mathcal{B}$ peut les générer lui même (la distance statistique entre les signatures de différents membres du cercle est négligeable).

2.5.1 Nouveau schéma

Afin de résoudre ce problème nous modifions l'algorithme de génération des clés. Chaque utilisateur génère un ensemble qui contient k clés de vérification, k étant le paramètre de sécurité. Parmi ces clés, il y a $k/2$ qui sont générées par l'algorithme de génération des clés $\mathsf{Ring-gen-keys}$, l'utilisateur sauvegarde les clés secrètes correspondantes. Les autres $k/2$ clés de vérification sont choisies de manière uniformément aléatoire dans l'espace des fonctions de hachage. Les k clés de vérification sont numérotées, l'ordre est choisi aléatoirement (en mélangeant les deux types de clés).

Remarque 2.8. *Pour un cercle R tel que $index(R) \subset [\ell]$ ($\ell \leq k^c$), nous notons par $\texttt{fulldesc}(R)$ la description de toutes les clés de vérification (publiques) des utilisateurs qui sont dans R.*

Lors de la signature d'un message μ par un membre du cercle R, l'utilisateur fait un appel à un oracle aléatoire avec $(\mu, \texttt{fulldesc}(R))$ comme requête. La réponse de l'oracle est l'ensemble $\{T_{\sigma,i}\}_{i \in index(R)}$, où $T_{\sigma,i}$ est un sous ensemble de $[k]$ de cardinal $k/2$ pour tout $i \in index(R)$.

Nous considérons le même algorithme de génération de paramètre $\mathsf{Ring-gen-params}$ que celui du schéma dans le section 2.3. Par conséquent, nous avons $\mathcal{P} \leftarrow \mathsf{Ring-gen-params}(1^k)$, avec $\mathcal{P} = (k, n, m_u, p, \mathbf{S})$. Soit R un cercle de taille $\ell \leq k^c$ tel que $index(R) = [\ell]$.

$\mathsf{Ring-gen-keys-ic}(\mathcal{P})$:

1. Soit i l'indice d'un utilisateur qui fait partie du cercle R.
2. Choisir aléatoirement un sous ensemble $T \subset [k]$ de taille $k/2$.
3. Pour $j \in T$, soient $pk_{i,j} \leftarrow \mathcal{H}(\mathcal{D}, D_h, m_u)$ et $sk_{i,j} = 0$.
4. Pour $j \in [k] \backslash T$, soit $(pk_{i,j}, sk_{i,j}) \leftarrow \mathsf{Ring-gen-keys}(\mathcal{P})$.
5. Retourner $(pk_i, sk_i) = (\{pk_{i,j}\}_{j \in [k]}, \{sk_{i,j}\}_{j \in [k]})$.

$\mathsf{Ring-sign-ic}(\mathcal{P}, sk_{i_0}, \mu, R)$: Nous supposons que l'utilisateur d'indice i_0 dans $R = \{pk_i\}_{i \in [\ell]}$ veut générer une signature de cercle pour un message $\mu \in \{0,1\}^\star$.

0. Vérifier que : les paramètres publics respectent les contraintes $1-3$ dans $\mathsf{Ring-gen-params}$; chaque $sk_{i_0,j} \in sk_{i_0}$ est dans $D_{s,c}^{m_u}$ pour tout $j \in [k]$; R est de taille ℓ inférieur à k^c ; une parmi les clés publiques dans R est associée à sk_{i_0}. Si la vérification échoue, alors retourner $\texttt{failed}$.
1. Soit $\{T_{\sigma,i}\}_{i \in [\ell]} \leftarrow H_\sigma(\mu, \texttt{fulldesc}(R))$

2. Soit $\texttt{keys}(R) = \{pk_{i,j}\}_{i\in[\ell],j\in T_{\sigma,i}}$
3. Choisir aléatoirement $sk_{i_0,x} \in sk_{i_0}$ avec $x \in T_{\sigma,i_0}$ telle que $sk_{i_0,x} \neq 0$. S'il n'existe aucun indice dans T_{σ,i_0} pour lequel on connait une clé secrète, abandonner.
4. Retourner $\sigma = (\{\hat{\mathbf{z}}_{i,j}\}_{i\in[\ell],j\in T_{\sigma,i}}, \mathbf{e}) \leftarrow \mathsf{Ring{-}sign}(\mathcal{P}, sk_{i_0,x}, \mu, \texttt{keys}(R))$ comme signature.

$\mathsf{Ring{-}verify{-}ic}(\mathcal{P}, \mu, R, \sigma)$: Étant donnés un message μ, un cercle R et une signature de cercle $\sigma = (\{\hat{\mathbf{z}}_{i,j}\}_{i\in[\ell],j\in T_{\sigma,i}}, \mathbf{e})$, le vérificateur commence par appeler l'oracle H_σ :
— Soit $\{T_{\sigma,i}\}_{i\in[\ell]} \leftarrow H_\sigma(\mu, \texttt{fulldesc}(R))$.
— Soit $\texttt{keys}(R) = \{pk_{i,j}\}_{i\in[\ell],j\in T_{\sigma,i}}$.
Le vérificateur accepte la signature si les conditions suivantes sont satisfaites :

1. $\hat{\mathbf{z}}_{i,j} \in D_z^{m_u}$ pour tous $i \in [\ell]$ et $j \in T_{\sigma,i}$.
2. $\mathbf{e} = H(\sum_{i\in[\ell],j\in T_{\sigma,i}} h_{i,j}(\hat{\mathbf{z}}_{i,j}){-}\mathbf{Se}, \texttt{keys}(R), \mu)$ où $\{h_{i,j}\}_{i\in[\ell],j\in T_{\sigma,i}} = \texttt{keys}(R)$.

Sinon, le vérificateur refuse la signature.

Remarque 2.9. *Puisque l'oracle aléatoire choisit $k/2$ clés de vérification de signataire aléatoirement, alors la probabilité qu'elles soient toutes des fonctions de hachage aléatoires est exponentiellement faible en k. Par conséquent la probabilité d'abandon dans l'étape (3) de l'algorithme* $\mathsf{Ring{-}sign{-}ic}$ *est négligeable.*

2.5.2 Anonymat

Dans le jeu d'anonymat inconditionnel contre les attaques à paramètres choisis, l'attaquant reçoit une signature qui dépend d'un bit aléatoire b ainsi qu'un ensemble de paramètres publics $\mathcal{P} = (k, n, m_u, p, \mathbf{S})$, deux clés secrètes sk_{i_0}, sk_{i_1} telles que pour $b \in \{0,1\}$, $sk_{i_b} = \{sk_{i_b,j}\}_{j\in[k]}$, un message μ et un cercle R de taille ℓ ($\ell \leq k^c$) tel que $index(R) = [\ell]$. Tous ces paramètres sont choisis par l'attaquant sauf le bit aléatoire b.

Soit $\tilde{X}_{b,\mathcal{P},sk_{i_b},\mu,R}$ la variable aléatoire qui représente la sortie de l'algorithme de signature $\mathsf{Ring{-}sign{-}ic}(\mathcal{P}, sk_{i_b}, \mu, R)$. Soient $\{T_{\sigma,i}\}_{i\in[\ell]} \leftarrow H_\sigma(\mu, \texttt{fulldesc}(R))$ et $\texttt{keys}(R) = \{pk_{i,j}\}_{i\in[\ell],j\in T_{\sigma,i}}$. Nous notons par $X_{b,\mathcal{P},sk_{i_b,x},\mu,\texttt{keys}(R)}$ la variable aléatoire représentant la sortie de l'algorithme de signature $\mathsf{Ring{-}Sign}(\mathcal{P}, sk_{i_b,x}, \mu, \texttt{keys}(R))$. Par conséquent, nous avons que $\tilde{X}_{b,\mathcal{P},sk_{i_b},\mu,R} = X_{b,\mathcal{P},sk_{i_b,x},\mu,\texttt{keys}(R)}$, pour $b \in \{0,1\}$. Pour le cercle de taille $\ell \cdot k/2$, défini par $\texttt{keys}(R)$, d'après le Théorème 2.3, nous avons que

$$\Delta(X_{0,\mathcal{P},sk_{i_0,x},\mu,\texttt{keys}(R)}, X_{1,\mathcal{P},sk_{i_1,x},\mu,\texttt{keys}(R)}) \leq n^{-\omega(1)}.$$

Ceci implique directement que le nouveau schéma de la section 2.5.1 satisfait l'anonymat inconditionnel contre les attaques à paramètres choisis.

2.5.3 Preuve d'inforgeabilité contre corruption interne

Nous utiliserons l'attaquant du jeu d'inforgeabilité contre corruption interne pour casser un challenge du schéma LSig09. Pour ce faire, nous considérons un vecteur de $m' = k/2 \cdot \ell \cdot m_u$ polynômes comme challenge. Nous utilisons ces polynômes pour notre schéma de signature de cercle au lieu des fonctions de hachage choisies uniformément pendant l'étape (3) de l'algorithme $\mathsf{Ring-gen-keys-ic}$. Le reste des clés sont générées grâce à l'algorithme $\mathsf{Ring-gen-keys}$. À noter que puisque la moitié des clés de vérification de chaque utilisateur ont été générées par cet algorithme, nous pouvons répondre à toutes les requêtes de corruption de l'attaquant.

Pour répondre au challenge, nous voulons que l'attaquant retourne une signature qui utilise comme clés de vérification celles qui correspondent au challenge et aucunes de celles générées par l'algorithme $\mathsf{Ring-gen-keys}$. L'idée principale est d'appeler l'attaquant avec un oracle aléatoire contrôlé H_σ, générer (de manière uniformément aléatoire) un nombre polynomial d'éléments dans l'espace d'arrivée de l'oracle H_σ et deviner quel élément parmi ceux-ci va être utilisé dans la réponse de l'attaquant. En suivant cette approche nous pouvons ordonner les clés des utilisateurs de telle sorte que si l'attaquant utilise la réponse de l'oracle que nous souhaitons, il choisira uniquement les clés que nous n'avons pas associées des clés de signature. Quand cela arrive (après un nombre polynomial d'appel de l'attaquant), nous obtenons une signature de la part de l'attaquant qui peut être utilisée pour répondre au challenge initial.

Théorème 2.6. *S'il existe un algorithme en temps polynomial qui peut gagner le jeu d'inforgeabilité contre corruption interne pour notre schéma de signature de cercle, alors il existe un algorithme en temps polynomial qui peut résoudre* $(x^n + 1)$-$\mathsf{SVP}_\gamma^\infty$ *dans le pire cas pour* $\gamma = \tilde{O}(n^{2.5+2c})$.

Démonstration. Pour démontrer le théorème nous utilisons la même idée dans la preuve du Théorème 2.5. Nous supposons que nous avons un attaquant $\mathcal{A}$ qui peut forger des signatures dans le jeu d'inforgeabilité contre corruption interne. En utilisant $\mathcal{A}$ nous construisons un attaquant $\mathcal{B}$ qui peut forger des signatures pour le schéma de signature LSig09.

Préparation : Au début, $\mathcal{B}$ reçoit comme challenge une description d'une fonction de hachage (un vecteur de $m' = k/2 \cdot \ell \cdot m_u$ polynômes) et un élément $\mathbf{S} \in \mathcal{D}$. Les $k/2$ clés restantes sont générées suivant l'algorithme $\mathsf{Ring-gen-keys}$. Ainsi $\mathcal{B}$ peut répondre aux requêtes de corruption venant de $\mathcal{A}$ puisque la moitié des clés de vérification sont générées par l'algorithme $\mathsf{Ring-gen-keys}$.

$\mathcal{B}$ génère un nombre polynomial d'éléments dans l'espace d'arrivée de l'oracle aléatoire H_σ et les enregistre dans une liste $E = \{Q_1, \cdots, Q_t\}$ avec $t = \mathsf{poly}(k)$. Chaque élément Q_w ($w \in [t]$) de l'ensemble E consiste en un ensemble $\{T_{\sigma,i}\}_{i \in I_w}$, avec $I_w \subseteq [k^c]$ et $T_{\sigma,i}$ est un sous ensemble de $[k]$ de cardinal $k/2$ pour tout $i \in I_w$. Puis,

$\mathcal{B}$ choisit au hasard un élément dans la liste E, nous le notons par Q_s avec $s \in [t]$ (Q_s est l'élément que l'attaquant $\mathcal{A}$ est supposé utiliser dans sa réponse). Nous supposons que $Q_s = \{T_{\sigma,i}\}_{i \in I_s}$ avec $I_s \subseteq [k^c]$ de cardinal $x_0 \leq \ell$.

Ensuite, $\mathcal{B}$ ordonne l'ensemble des $k \cdot \ell \cdot m_u$ polynômes de telle sorte que Q_s correspond à des clés publiques qui font partie de notre challenge, autrement dit avec des polynômes du vecteur m'. Nous supposons que les polynômes dans m' sont ordonnées et que $\mathcal{B}$ utilise les $k/2 \cdot x_0 \cdot m_u$ premiers polynômes pour les clés publiques dans Q_s. Finalement, $\mathcal{B}$ commence le jeu par initialiser $\mathcal{A}$ en lui donnant l'ensemble de toutes les clés publiques. Nous notons $S = \{pk_i\}_{i \in index(S)}$ le cercle formé par ces clés publiques, c'est un cercle contenant exactement ℓ utilisateurs (le cardinal de $index(S)$ est égal à ℓ).

Interaction avec les oracles : Nous rappelons que $\mathcal{B}$ doit répondre aux requêtes de corruption, requêtes de signature, requêtes pour l'oracle H_σ et les requêtes pour l'oracle H utilisé par l'algorithme de signature $\mathsf{Ring-sign}$ (voir l'étape (4) de l'algorithme $\mathsf{Ring-sign-ic}$). Nous expliquons dans ce qui suit comment $\mathcal{B}$ peut répondre à ces différents requêtes :

— *Requête de corruption* : Pour chaque requête de corruption, $\mathcal{B}$ répond avec l'ensemble sk qui contient les $k/2$ clés secrètes valides (clés générées par l'algorithme $\mathsf{Ring-gen-keys}$).

— *Requête pour l'oracle H* : Pour chaque requête (x_y, x_h, x_m) à l'oracle aléatoire H, il teste s'il a déjà répondu à une telle requête. Si c'est le cas, il répond toujours avec le même résultat. Sinon, il répond par $H_L(x_y, x_h \| x_m)$, et il enregistre le résultat dans une liste, où H_L est l'oracle aléatoire utilisé dans le schéma LSig09.

— *Requête pour l'oracle H_σ* : Une requête pour l'oracle H_σ est de la forme $(\mu, \texttt{fulldesc}(R))$, où μ est un message quelconque dans $\{0,1\}^\star$ et R est un cercle de taille inférieur à k^c. Pour répondre à cette requête, $\mathcal{B}$ cherche un élément $Q_w = \{T_{\sigma,i}\}_{i \in I_w}$ dans la liste E tel que $I_w = index(R)$. S'il ne trouve aucun élément alors il abandonne le jeu.

— *Requête de signature* : Les requêtes pour l'oracle de signature sont de la forme (i_0, μ, R) avec R un cercle de taille x_1 inférieur à k^c, μ un message dans $\{0,1\}^\star$ et i_0 est l'indice du signataire tel que $i_0 \in index(S) \cap index(R)$. $\mathcal{B}$ suit les étapes suivantes pour répondre à cette requête :

1. Choisir $Q_w = \{T_{\sigma,i}\}_{i \in I_w}$ dans la liste E tel que $I_w = index(R)$. S'il ne trouve aucun élément alors il abandonne le jeu.
2. Il programme l'oracle H_σ de sorte que $H_\sigma(\mu, \texttt{fulldesc}(R)) = \{T_{\sigma,i}\}_{i \in I_w}$.
3. Soit $\texttt{keys}(R) = \{pk_{i,j}\}_{i \in I_w, j \in T_{\sigma,i}}$ où $T_{\sigma,i} \in Q_w$ pour tout i.
4. Choisir aléatoirement $sk_{i_0,x} \in sk_{i_0}$ où $x \in T_{\sigma,i_0}$ telle que $sk_{i_0,x} \neq 0$. S'il

n'existe aucun indice dans T_{σ,i_0} pour lequel on connait une clé secrète, abandonner.

5. Il exécute l'algorithme $\mathsf{Ring-sign}$ avec $(\mathcal{P}, sk_{i_0,x}, \mu, \mathsf{keys}(R))$ comme entrée. Soit $\sigma = (\{\hat{\mathbf{z}}_{i,j}\}_{i\in I_w, j\in T_{\sigma,i}}, \mathbf{e})$ la signature obtenue.
6. Il vérifie si H_L a été appelé avec des paramètres

$$(\sum_{i\in I_w, j\in T_{\sigma,i}} h_{i,j}(\hat{\mathbf{z}}_{i,j}) - \mathbf{Se}, \mathtt{keys}(R)\|\mu),$$

où $h_{i,j} \in \mathtt{keys}(R)$ pour tous i et j. Si c'est le cas il abandonne le jeu, sinon il retourne $\sigma = (\{\hat{\mathbf{z}}_{i,j}\}_{i\in I_w, j\in T_{\sigma,i}}, \mathbf{e})$ comme signature.

Génération de la signature : À un moment donné, l'attaquant $\mathcal{A}$ retourne avec une probabilité non négligeable une signature $(\{\hat{\mathbf{z}}_{i,j}\}_{i\in I_s, j\in T_{\sigma,i}}, \mathbf{e}, \mu)$ où $\{T_{\sigma,i}\}_{i\in I_s} = Q_s$, nous notons ce cercle par T. Comme dans la preuve du Théorème 2.5, nous montrons que $\mathcal{B}$ peut exploiter la réponse de $\mathcal{A}$ pour casser l'inforgeabilité du schéma LSig09 en utilisant une fonction de hachage définie par les polynômes du challenge $m' = k/2 \cdot \ell \cdot m_u$.

Dans le cas où l'attaquant n'utilise pas l'élément Q_s choisi au départ, $\mathcal{B}$ relance l'attaquant $\mathcal{A}$ jusqu'à l'obtention de la réponse souhaitée.

Analyse : Dans cette partie nous allons reprendre les arguments utilisés dans la preuve du Théorème 2.5. Dans un premier temps, nous allons montrer que toutes les entrées reçues par $\mathcal{A}$ ont une distribution statistiquement proche de celle des éléments qu'il aurait reçus dans un challenge de la signature de cercle. Ensuite, nous montrons que la signature obtenue par $\mathcal{B}$ est valide pour le schéma LSig09.

Tout d'abord, nous avons que la distribution du challenge donné à $\mathcal{A}$ au début du jeu est statistiquement proche de celle des challenges de la signature de cercle. En effet, nos avons que la moitié des clés ($m' = k/2 \cdot \ell \cdot m_u$) sont générées de manière uniformément aléatoire. L'autre moitié est générée suivant l'algorithme $\mathsf{Ring-gen-keys}$ avec des clés secrètes valides. D'autre part, la distribution des signatures générées par $\mathcal{B}$ pendant la phase de requête, est statistiquement proche de celles obtenues à partir de l'algorithme $\mathsf{Ring-sign-ic}$. En effet, puisque nous connaissons toutes les clés secrètes, nous n'avons pas besoin de programmer l'oracle aléatoire comme nous l'avons fait dans la preuve du Théorème 2.5.

En suivant la preuve du Théorème 2.5, nous utilisons le fait que le vecteur $\mathbf{e}$ de la signature $(\{\hat{\mathbf{z}}_{i,j}\}_{i\in I_s, j\in T_{\sigma,i}}, \mathbf{e}, \mu)$ de l'attaquant $\mathcal{A}$ est généré par un appel à l'oracle aléatoire pour montrer que $\mathcal{B}$ casse l'inforgeabilité du schéma LSig09. Donc nous avons deux cas : ou bien $\mathbf{e}$ a été généré durant une requête de signature, ou bien $\mathbf{e}$ est une réponse à un appel direct de l'oracle aléatoire H.

Supposons que $\mathbf{e}$ a été généré durant une requête de signature qui a donnée comme résultat $(\{\hat{\mathbf{z}}'_{i,j}\}_{i\in I_w, j\in T'_{\sigma,i}}, \mathbf{e}, \mu')$ pour un cercle R' de taille x_2 tel que

$index(R') = I_w$. Pour être valide, la signature de $\mathcal{A}$ doit être différente de

$$(\{\hat{\mathbf{z}}'_{i,j}\}_{i\in I_w, j\in T'_{\sigma,i}}, \mathbf{e}, \mu', \mathtt{keys}(R')).$$

Par conséquent, nous aurons soit $\mathtt{keys}(R')\|\mu' \neq \mathtt{keys}(T)\|\mu$ ou bien $(\{\hat{\mathbf{z}}'_{i,j}\}_{i\in I_w, j\in T'_{\sigma,i}}) \neq (\{\hat{\mathbf{z}}_{i,j}\}_{i\in I_s, j\in T_{\sigma,i}})$, si $\mathtt{keys}(R')\|\mu' \neq \mathtt{keys}(T)\|\mu$ alors $\mathtt{keys}(R') \neq \mathtt{keys}(T)$ ou bien $\mu' \neq \mu$. Dans les deux cas nous avons une collision sur H puisque nous avons une seule image par H de deux pré-images qui sont différentes soit dans la deuxième ou dans la troisième coordonnée.

$$H(\sum_{i\in I_w, j\in T'_{\sigma,i}} h_{i,j}(\hat{\mathbf{z}}'_{i,j}) - \mathbf{Se}, \mathtt{keys}(R'), \mu') = H(\sum_{i\in I_s, j\in T_{\sigma,i}} h_{i,j}(\hat{\mathbf{z}}_{i,j}) - \mathbf{Se}, \mathtt{keys}(T), \mu) = \mathbf{e},$$

Dans le deuxième cas, en utilisant le même raisonnement, nous avons

$$\sum_{i\in I_w, j\in T'_{\sigma,i}} h_{i,j}(\hat{\mathbf{z}}'_{i,j}) - \mathbf{Se} = \sum_{i\in I_s, j\in T_{\sigma,i}} h_{i,j}(\hat{\mathbf{z}}_{i,j}) - \mathbf{Se} \text{ pour } (\{\hat{\mathbf{z}}'_{i,j}\}_{i\in I_w, j\in T'_{\sigma,i}}) \neq (\{\hat{\mathbf{z}}_{i,j}\}_{i\in I_s, j\in T_{\sigma,i}}).$$

Pour simplifier les notations et sans perte de généralité, nous supposons que $x_2 = x_0$ (x_2 est la taille du cercle R' et x_0 est la taille du cercle T) et que les cercles R' et T sont distincts (aucun utilisateur ne fait partie des deux cercles à la fois). En fixant $\hat{\mathbf{z}}_{i,j} = 0$ pour $i \in I_w, j \in T'_{\sigma,i}$ et $\hat{\mathbf{z}}'_{i,j} = 0$ pour $i \in I_s, j \in T_{\sigma,i}$, nous obtenons

$$\sum_{i\in I_w\cup I_s, j\in T'_{\sigma,i}\cup T_{\sigma,i}} h_{i,j}(\hat{\mathbf{z}}'_{i,j}) = \sum_{i\in I_s\cup I_w, j\in T_{\sigma,i}\cup T'_{\sigma,i}} h_{i,j}(\hat{\mathbf{z}}_{i,j}).$$

Ainsi, nous obtenons une collision pour une fonction de hachage dans $\mathcal{H}(\mathcal{D}, D_h, k/2\cdot 2x_2 \cdot m_u)$, d'après le Théorème 2.1, nous déduisons que nous pouvons résoudre le problème (x^n+1)-SVP$^{\infty}_{\gamma}$, pour $\gamma = \tilde{O}(n^{2.5+2c})$.

Passons maintenant à le cas où $\mathbf{e}$ a été généré par une requête directe à l'oracle aléatoire. Nous avons $(\{\hat{\mathbf{z}}_{i,j}\}_{i\in I_s, j\in T_{\sigma,i}}, \mathbf{e}, \mu)$ comme signature forgée, d'après l'algorithme de vérification Ring$-$verify$-$ic, nous avons que

$$H(\sum_{i\in I_s, j\in T_{\sigma,i}} h_{i,j}(\hat{\mathbf{z}}_{i,j}) - \mathbf{Se}, \mathtt{keys}(T), \mu) = \mathbf{e}.$$

En notant $x_y = \sum_{i\in I_s, j\in T_{\sigma,i}} h_{i,j}(\hat{\mathbf{z}}_{i,j}) - \mathbf{Se}$ et compte tenu de l'algorithme utilisé par $\mathcal{B}$, nous avons aussi $H_L(x_y, \mathtt{keys}(T)\|\mu) = \mathbf{e}$. Nous notons par $h'_1, \cdots, h'_{(\ell-x_0)\cdot k/2}$ les fonctions de hachage correspondantes aux clés publiques du challenge initial m' qui n'ont pas été sélectionnées pendant l'étape de préparation pour faire partie de la réponse Q_s. Pour ces fonctions nous choisissons $(\ell - x_0)\cdot k/2$ vecteurs nuls que nous notons par $\hat{\mathbf{z}}'_1, \cdots, \hat{\mathbf{z}}'_{(\ell-x_0)\cdot k/2}$. Par conséquent, nous obtenons que

$$\sum_{i\in I_s, j\in T_{\sigma,i}} h_{i,j}(\hat{\mathbf{z}}_{i,j}) + \sum_{i\in[(\ell-x_0)\cdot k/2]} h'_i(\hat{\mathbf{z}}'_i) - \mathbf{Se} = x_y,$$

ce qui implique que

$$H_L(\sum_{i\in I_s, j\in T_{\sigma,i}} h_{i,j}(\hat{\mathbf{z}}_{i,j}) + \sum_{i\in[(\ell-x_0)\cdot k/2]} h'_i(\hat{\mathbf{z}}'_i) - \mathbf{Se}, \mathtt{keys}(T)\|\mu) = \mathbf{e}.$$

La concaténation de tous les vecteurs de polynômes (dans l'ordre) $h_{i,j}$ pour $i \in I_s, j \in T_{\sigma,i}$ avec le vecteur $h'_1\| \cdots \|h'_{(\ell-x_0)\cdot k/2}$ permet de reconstruire le vecteur de polynômes challenge $m' = k/2 \cdot \ell \cdot m_u$. Soit $\hat{\mathbf{z}} \in D_z^{k/2\cdot\ell\cdot m_u}$ le vecteur obtenu par la concaténation (dans l'ordre) des vecteurs $\hat{\mathbf{z}}_{i,j}$ pour $i \in I_s, j \in T_{\sigma,i}$ et le vecteur $\hat{\mathbf{z}}'_1\| \cdots \|\hat{\mathbf{z}}'_{(\ell-x_0)\cdot k/2}$. Donc, nous obtenons $\sigma' = (\hat{\mathbf{z}}, \mathbf{e})$ est une signature de $\mu' = \mathtt{keys}(T)\|\mu$ et comme nous n'avons pas fait de requête de signature pour le schéma LSig09 alors (σ', μ') est une signature forgée valide pour une fonction de hachage qui correspond à notre challenge m'.

Pour finir la preuve, nous nous intéressons à la probabilité d'abandon dans les différents étapes. En utilisant la Remarque 2.9, nous avons que la probabilité d'abandon dans l'étape (4) de l'algorithme de génération de signature utilisé par $\mathcal{B}$ est négligeable. D'autre part, la probabilité d'abandon dans l'étape (6) est négligeable aussi. En effet, d'après le Lemme 2.7, si on fait d'augmenter le nombre de fonctions de hachage utilisées, la quantité $\sum_{i\in I_w, j\in T_{\sigma,i}} h_{i,j}(\hat{\mathbf{z}}_{i,j}) - \mathbf{Se}$ est à une distance négligeable de la distribution uniform sur $\mathcal{D}$. Par conséquent, la probabilité que $\mathcal{A}$ génère une requête en collision avec celle de l'algorithme de signature de la preuve est négligeable.

Deux autres possibilités d'abandon se produisent au niveau de la phase de requête pour l'oracle H_σ et de la phase de requête de signature (étape (1)). Dans les deux cas, nous avons le même problème $\mathcal{B}$ ne trouve pas la bonne réponse dans la liste initiale E. Dans ces deux cas, on a pas le choix il faut relancer l'attaquant $\mathcal{A}$ de nouveau. □

Chapitre 3

Schémas d'identification anonymes

Sommaire

Chapitre 3

Schémas d'identification anonymes

Nous allons nous intéresser dans ce chapitre aux schémas permettant l'identification anonyme d'un ou plusieurs utilisateurs. Dans la section 3.1, nous proposons un nouveau schéma d'identification de cercle. Cette notion a été introduite en 2004 par Dodis, Kiayias, Nicolosi, et Shoup dans [DKNS04], elle permet à un utilisateur de prouver son appartenance à une entité sans révéler son identité. Nous comparons les performances de notre schéma avec celles d'un schéma proposé par Kawachi, Tanaka, et Xagawa [KTX08]. Dans la section 3.2, nous généralisons ce résultat au cas où un ensemble d'utilisateurs veut donner une preuve d'appartenance à une entité d'une manière anonyme. Nous comparons les performances de notre schéma avec un schéma proposé par Cayrel, Lindner, Rückert et Silva [CLRS10b].

Une grande partie des travaux que nous présentons dans ce chapitre est issue d'une collaboration avec Julien Schrek ayant donné lieu à une publication intitulée *Improved Lattice-Based Threshold Ring Signature Scheme* [BS13] à la conférence PQCrypto 2013.

3.1 Schémas d'identification de cercle

Dans [KTX08], Kawachi, Tanaka et Xagawa ont proposé un schéma d'identification de cercle dont la sécurité est basée sur les problèmes SIS (voir la Définition 1.27) et ISIS (voir la Définition 1.28). Le schéma est une variation du schéma d'identification en trois passes présenté dans la section 1.6.3. Leur construction hérite la probabilité de fraude de $2/3$ du schéma d'identification de base, par conséquent le protocole doit être exécuter un nombre de fois $t_1 = \omega(\log n)$ où n est le paramètre de sécurité, afin d'obtenir une probabilité de triche négligeable. Le coût de communication du protocole est de $t_1 \cdot \tilde{O}(n+\ell)$, avec ℓ la taille de cercle auquel appartient le prouveur.

Dans la section 3.1.3, nous présentons une amélioration du schéma proposé

par Kawachi et al. [KTX08]. Notre schéma est une variation du schéma d'identification en cinq passes présenté en Annexe A. Le schéma est parallélisable avec une probabilité de triche $1/2$ et donc plus faible que celle dans [KTX08].

3.1.1 Définitions et modèle de sécurité

Nous utiliserons le formalisme des schémas d'identification de cercle proposé par Kawachi et al. [KTX08] et Xagawa [Xag10] pour décrire les définitions et le modèle de sécurité.

Schéma d'identification de cercle. Un schéma d'identification de cercle consiste en le quadruplet d'algorithmes $(\mathsf{Ring{-}id{-}init}, \mathsf{Ring{-}id{-}keygen}, \mathsf{Ring{-}id{-}ccp}, \mathsf{Ring{-}id{-}ccs})$:

- $\mathsf{param} \leftarrow \mathsf{Ring{-}id{-}init}(1^n)$: $\mathsf{Ring{-}id{-}init}$ est un algorithme probabiliste polynomial d'initialisation, il prend en entrée un paramètre de sécurité 1^n et retourne un paramètre public param.
- $(pk_i, sk_i) \leftarrow \mathsf{Ring{-}id{-}keygen}(\mathsf{param}, i)$: $\mathsf{Ring{-}id{-}keygen}$ est un algorithme probabiliste polynomial qui prend en entrée le paramètre param et l'identité i d'un utilisateur et retourne une paire de clés (pk_i, sk_i).
- $\mathsf{Ring{-}pk} \leftarrow \mathsf{Ring{-}id{-}ccp}(\mathsf{param}, R = (pk_1, \cdots, pk_\ell))$: $\mathsf{Ring{-}id{-}ccp}$ est un algorithme qui permet de générer la clé publique du cercle. L'algorithme prend en entrée le paramètre public param et le cercle défini par $R = (pk_1, \cdots, pk_\ell)$ et retourne la clé publique du cercle $\mathsf{Ring{-}pk}$.
- $\mathsf{Ring{-}sk} \leftarrow \mathsf{Ring{-}id{-}ccs}(\mathsf{param}, R = (pk_1, \cdots, pk_\ell), sk_i)$: est un algorithme qui permet de générer la clé secrète du cercle. L'algorithme prend en entrée param, un cercle de taille ℓ défini par $R = (pk_1, \cdots, pk_\ell)$ et une clé secrète sk_i qui correspond à une clé publique $pk_i \in R$.
- $(\mathcal{P}(\mathsf{Ring{-}sk}) \leftrightarrow \mathcal{V})(\mathsf{param}, \mathsf{Ring{-}pk})$: est un algorithme interactif entre un prouveur $\mathcal{P}$ qui possède la clé secrète du cercle $\mathsf{Ring{-}sk}$ et un vérificateur $\mathcal{V}$. Les deux protagonistes partagent le paramètre public param et la clé publique du cercle $\mathsf{Ring{-}pk}$. A la fin de l'interaction, le vérificateur retourne $\mathsf{desc} \in \{0, 1\}$, qui vaut 1 si ce dernier accepte et 0 s'il rejette.

Nous avons la définition de complétude suivante :

Définition 3.1 (Complétude). *Pour tous n, ℓ (polynomial en n),* $\mathsf{param} \in \mathsf{Ring{-}id{-}init}(1^n)$, $\{(pk_i, sk_i)\}_{i\in[\ell]} \subset \mathsf{Ring{-}id{-}keygen}(\mathsf{param})$, $\mathsf{Ring{-}pk} \subset \mathsf{Ring{-}id{-}ccp}(\mathsf{param}, \{pk_i\}_{i\in[\ell]})$, $i_0 \in [\ell]$ *et* $\mathsf{Ring{-}sk} \subset \mathsf{Ring{-}id{-}ccs}(\mathsf{param}, \{pk_i\}_{i\in[\ell]}, sk_{i_0})$ *on a,*

$$\Pr\left[\mathsf{desc} = 1 : \mathsf{desc} \leftarrow (\mathcal{P}(\mathsf{Ring{-}sk}) \leftrightarrow \mathcal{V})(\mathsf{param}, \mathsf{Ring{-}pk})\right] = 1.$$

3.1.1.1 Modèle de sécurité

Un schéma d'identification de cercle doit être sûr contre l'usurpation d'identité et il doit être anonyme.

Usurpation d'identité : Dans [KTX08], Kawachi et al. ont introduit la notion d'usurpation d'identité avec des attaques parallèles à cercle choisi. Dans le jeu qui modélise cette notion, il existe deux protagonistes, un challengeur et un attaquant. Le but de l'attaquant est de se faire passer pour un utilisateur arbitraire dans un cercle choisi par lui même.

L'attaquant $\mathcal{I}$ (en anglais *impersonator*) est modélisé par un couple d'algorithmes interactifs $(\tilde{\mathcal{P}}, \tilde{\mathcal{V}})$ qui sont exécutés pendant les différentes phases du jeu formalisant la notion d'usurpation d'identité. Nous résumons les moyens mis à disposition de $\mathcal{I}$ durant ce jeu dans les points suivants :

1. En utilisant $\tilde{\mathcal{V}}$ (vérificateur malhonnête), l'attaquant peut forcer un prouveur honnête (simulé par le challengeur) à prouver son appartenance à un cercle de son choix contenant le prouveur.
2. En utilisant $\tilde{\mathcal{V}}$ (vérificateur malhonnête), l'attaquant peut exécuter plusieurs instances du protocole en parallèle avec des clones de n'importe quel prouveur.
3. Pendant la phase de challenge, l'attaquant définit un cercle challenge. En utilisant $\tilde{\mathcal{P}}$ (prouveur malhonnête), l'attaquant peut interagir avec des prouveurs de la même façon que $\tilde{\mathcal{V}}$ dans (1) et (2), mais il est interdit d'interagir avec des prouveurs qui font partie de son cercle challenge. Il peut obtenir des clés secrètes des prouveurs mais il n'a pas le droit d'avoir celles qui correspondent aux utilisateurs appartenant à son cercle challenge.

Dans la suite de ce chapitre, nous utiliserons les notations suivantes :
— UH décrira l'ensemble des utilisateurs honnêtes.
— UC décrira l'ensemble des utilisateurs corrompus par l'attaquant.
— US décrira l'ensemble des utilisateurs cible (ou bien challenge).

Maintenant nous donnons la définition formelle de la notion d'usurpation d'identité avec des attaques parallèles à cercle choisi, suivant [Xag10], [KTX08].

Définition 3.2. *Soit un schéma d'identification de cercle défini par les algorithmes (*$\mathsf{Ring-id-init}, \mathsf{Ring-id-keygen}, \mathsf{Ring-id-ccp}, \mathsf{Ring-id-ccs}$*). On considère le jeu suivant entre un challengeur $\mathcal{C}$ et un attaquant $\mathcal{I}$.*

Phase de préparation : *Soit n le paramètre de sécurité. Le challengeur initialise les ensembles UH, UC, et US à l'ensemble vide, ensuite il envoie* $(1^n, \mathsf{param} \leftarrow \mathsf{Ring-id-init}(1^n))$ *à l'attaquant.*

Phase d'apprentissage : *Pendant cette phase l'attaquant $\tilde{\mathcal{V}}$ peut envoyer des requêtes aux trois oracles suivants :*

— Init. *Il prend en entrée* i *et retourne* $\perp$ *si* $i \in UH \cup UC \cup US$. *Sinon il utilise l'algorithme de génération des clés pour obtenir* $(pk_i, sk_i) \leftarrow \mathsf{Ring-id-keygen}(\mathsf{param}, i)$ *puis il rajoute* i *à l'ensemble* UH *et il retourne la clé publique* pk_i.

— Corrupt. *Il prend* i *comme entrée et retourne* $\perp$ *si* $i \notin UH \setminus US$. *Sinon il rajoute* i *à l'ensemble* UC *puis il supprime* i *de l'ensemble* UH *et il retourne la clé secrète* sk_i *qui correspond à l'utilisateur* i.

— Prouv. *Il prend en entrée* $R = (pk_{i_1}, \cdots, pk_{i_l}), i_t$ *et* μ_{in}. *Si* $pk_{i_t} \notin R$ *ou bien* $i_t \notin UH \setminus US$ *alors retourne* $\perp$. *Sinon il utilise les algorithmes de génération des clés du cercle correspondants à l'utilisateur* i_t, *par conséquent il obtient* $\mathsf{Ring-pk} \leftarrow \mathsf{Ring-id-ccp}(\mathsf{param}, R)$ *et* $\mathsf{Ring-sk} \leftarrow \mathsf{Ring-id-ccs}(\mathsf{param}, R, sk_{i_t})$. *Puis il initialise l'état du prouveur par* $st_{\mathcal{P}} \leftarrow (\mathsf{param}, R, \mathsf{Ring-pk}, \mathsf{Ring-sk}, sk_{i_t})$. *Finalement il retourne* $\mu_{out} \leftarrow \mathcal{P}(\mu_{in}, st_{\mathcal{P}})$.

Phase de challenge : *Dans cette phase* $\tilde{\mathcal{V}}$ *envoie à* $\mathcal{C}$ *un challenge* $R_c = (pk_{i_1}, \cdots, pk_{i_c})$. *Si* $\{i_1, \cdots, i_c\} \nsubseteq UH$, *alors le challengeur abandonne le jeu. Sinon,* $\mathcal{C}$ *initialise* $US \leftarrow \{i_1, \cdots, i_c\}$ *et commence la phase d'usurpation d'identité avec l'attaquant* $\tilde{\mathcal{P}}$. *Ce dernier peut faire des requêtes aux oracles* Init, Corrupt, *et* Prouv. *A certain moment, le challengeur obtient*

$$(\mathsf{T}, \mathsf{desc}) \leftarrow \mathsf{Run}[(\tilde{\mathcal{P}}^{\mathsf{Init},\mathsf{Corrupt},\mathsf{Prouv}} \leftrightarrow \mathcal{V})(\mathsf{param}, R_c)],$$

avec $\tilde{\mathcal{P}}^{\mathsf{Init},\mathsf{Corrupt},\mathsf{Prouv}}$ *désigne un prouveur* $\tilde{\mathcal{P}}$ *ayant accès aux oracles* Init, Corrupt *et* Prouv.

L'attaquant $\mathcal{I} = (\tilde{\mathcal{P}}, \tilde{\mathcal{V}})$ *gagne le jeu si* $\mathsf{desc} = 1$. *Un schéma d'identification de cercle est sûr contre l'usurpation d'identité avec des attaques parallèles à cercle choisi, si la probabilité qu'un attaquant* $\mathcal{I}$ *puisse gagner ce jeu est négligeable en le paramètre de sécurité* n.

Anonymat : L'anonymat permet de s'assurer que l'on ne peut pas connaître les identités des prouveurs à partir de leurs transcriptions quand ils exécutent le protocole. Nous adopterons la notion d'anonymat contre la révélation totale des clés comme elle a été définie dans [Xag10], [KTX08]. Dans ce modèle, l'adversaire peut entrer en coalition avec tout les prouveurs et récupère leurs clés secrètes. Pour renforcer cette notion Kawachi et al. supposent que l'adversaire peut générer lui même les clés des prouveurs.

Un schéma d'identification de cercle est anonyme contre la révélation totale des clés si aucun attaquant ne peut distinguer entre deux prouveurs en regardant leurs transcriptions. Nous donnons une formalisation de cette notion dans la définition suivante.

Définition 3.3. *Soit un schéma d'identification de cercle défini par les algorithmes (*Ring−id−init, Ring−id−keygen, Ring−id−ccp, Ring−id−ccs*). On considère le jeu suivant entre un challengeur* $\mathcal{C}$ *et un attaquant* $\mathcal{A}$.

Phase de préparation : *Soit n le paramètre de sécurité. Le challengeur envoie à l'attaquant* $(1^n, \mathsf{param} \leftarrow \mathsf{Ring{-}id{-}init}(1^n))$.

Phase de challenge : *Dans cette phase $\mathcal{A}$ envoie à $\mathcal{C}$ un challenge* $(R, (pk_{i_0}, sk_{i_0}), (pk_{i_1}, sk_{i_1}))$ *tel que* $R = \{pk_i\}_{i\in[\ell]}$, $\{pk_{i_0}, pk_{i_1}\} \in R$ *et* $(pk_{i_0}, sk_{i_0}), (pk_{i_1}, sk_{i_1})$ *sont deux paires de clés valides. Ensuite le challengeur choisit un bit aléatoire* $b \in \{0,1\}$ *et il exécute le protocole comme étant un prouveur en utilisant la clé secrète* sk_{i_b}.

Finalement l'attaquant retourne un bit b' et il gagne le jeu si $b' = b$.

Nous définissons l'avantage de l'attaquant $\mathcal{A}$ comme étant

$$\mathsf{Adv}_{\mathcal{A}} = |\mathsf{Pr}[b' = b] - \frac{1}{2}|.$$

Un schéma d'identification de cercle satisfait la propriété d'anonymat contre la révélation totale des clés si tout attaquant $\mathcal{A}$ a un avantage $\mathsf{Adv}_{\mathcal{A}}$ négligeable en le paramètre de sécurité n.

3.1.2 Schéma d'identification de cercle de Kawachi-Tanaka-Xagawa

Le protocole d'identification de cercle proposé par Kawachi et al dans [KTX08] est une variation du protocole d'identification KTX.

Soient n le paramètre de sécurité et q, m deux entiers tels que $q = \mathsf{poly}(n)$ et $m \geq 10n \log q$. Le protocole est paramétré par une matrice $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$ choisie de manière uniformément aléatoire et rendue publique. Chaque prouveur i dans le cercle admet une paire de clés : une clé secrète qui consiste en un vecteur $\boldsymbol{x}_i \in \{0,1\}^m$ tel que $w_H(\boldsymbol{x}_i) = m/2$ et une clé publique qui correspond à un vecteur $\boldsymbol{y}_i \in \mathbb{Z}_q^n$ tel que $\boldsymbol{y}_i = \boldsymbol{A}\boldsymbol{x}_i \bmod q$.

Supposons que nous avons un cercle $R = \{\boldsymbol{y}_1, \cdots, \boldsymbol{y}_\ell\}$ de taille ℓ où chaque vecteur $\boldsymbol{y}_i$ correspond à la clé publique d'un utilisateur i. Soit $\mathcal{P}$ un prouveur ayant une paire de clés $(pk = \boldsymbol{y}_i, sk = \boldsymbol{x}_i)$, tel que $\boldsymbol{y}_i \in R$. Nous supposons que $\mathcal{P}$ veut s'identifier en tant qu'un membre du cercle R auprès d'un vérificateur $\mathcal{V}$. Pour cela, $\mathcal{P}$ donne une preuve de connaissance sans divulgation d'un secret $\boldsymbol{x}'$ tel que :

1. $\boldsymbol{x}'$ est de la forme $\boldsymbol{x}_i \| - \boldsymbol{e}_i$, où $\boldsymbol{x}_i \in \{0,1\}^m$, $w_H(\boldsymbol{x}_i) = m/2$, $\boldsymbol{e}_i \in \{0,1\}^\ell$ et $w_H(\boldsymbol{e}_i) = 1$.
2. $[\boldsymbol{A}\|\boldsymbol{y}_1\| \cdots \|\boldsymbol{y}_\ell]\boldsymbol{x}' = 0 \bmod q$, avec $\{\boldsymbol{y}_i\}_{i\in[\ell]} \in R$.

Afin de forcer le prouveur à démontrer qu'il connait un secret $\boldsymbol{x}'$ ayant la bonne forme, Kawachi et al ont modifié la permutation π utilisée dans le schéma d'identification de base KTX. La nouvelle permutation est définie grâce à deux permutations $\pi_m \in S_m$ et $\pi_\ell \in S_\ell$ et noté par $\pi = \pi_m \times \pi_\ell$ avec

$$\pi = \begin{pmatrix} 1 & 2 & \cdots & m \\ \pi_m(1) & \pi_m(2) & \cdots & \pi_m(m) \end{pmatrix} \cdot \begin{pmatrix} m+1 & 2 & \cdots & \ell \\ m+\pi_\ell(1) & m+\pi_\ell(2) & \cdots & m+\pi_\ell(\ell) \end{pmatrix}.$$

Nous avons les deux propriétés suivantes :
— $(\pi_m \times \pi_\ell)^{-1} = \pi_m^{-1} \times \pi_\ell^{-1}$, pour tout $\pi_m \in S_m$ et $\pi_\ell \in S_\ell$.
— Pour tout $\boldsymbol{a} \in \mathbb{Z}^m$ et $\boldsymbol{b} \in \mathbb{Z}^\ell$, si $\pi = \pi_m \times \pi_\ell$ alors $\pi(\boldsymbol{a}\|\boldsymbol{b}) = \pi_m(\boldsymbol{a})\|\pi_\ell(\boldsymbol{b})$.

Nous donnons maintenant une description formelle du schéma d'identification de cercle de Kawachi et al. [KTX08].

$\boldsymbol{A} \leftarrow \mathsf{Ring{-}id{-}init}(1^n)$: Étant donné un paramètre de sécurité n. Choisir une matrice $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$ d'une façon uniformément aléatoire.

$(\boldsymbol{y}_i, \boldsymbol{x}_i) \leftarrow \mathsf{Ring{-}id{-}keygen}(\boldsymbol{A}, i)$: Étant donnés une matrice $\boldsymbol{A}$ et un entier i qui correspond à l'indice de l'utilisateur dans le cercle, choisir une paire de clés en suivant la procédure suivante :
— Soit $\boldsymbol{x}_i \leftarrow \{0,1\}^m$, tel que $w_H(\boldsymbol{x}_i) = m/2$.
— Soit $\boldsymbol{y}_i = \boldsymbol{A}\boldsymbol{x}_i \bmod q$.

$\boldsymbol{A}' \leftarrow \mathsf{Ring{-}id{-}ccp}(\boldsymbol{A}, R = (\boldsymbol{y}_1, \cdots, \boldsymbol{y}_\ell))$: La clé publique du cercle R est une matrice $\boldsymbol{A}' \in \mathbb{Z}_q^{n\times(m+\ell)}$, telle que $\boldsymbol{A}' = [\boldsymbol{A}\|\boldsymbol{y}_1\|\cdots\|\boldsymbol{y}_\ell]$.

$\boldsymbol{x}' \leftarrow \mathsf{Ring{-}id{-}ccs}(\boldsymbol{A}, R = (\boldsymbol{y}_1, \cdots, \boldsymbol{y}_\ell), \boldsymbol{x}_i)$: La clé secrète du cercle est construite en suivant la procédure suivante :
— Soit $\boldsymbol{e}_i \in \{0,1\}^\ell$ le vecteur ayant 1 comme i-ème coordonnée et $w_H(\boldsymbol{e}_i) = 1$.
— Soit $\boldsymbol{x}'_i = \boldsymbol{x}_i\| - \boldsymbol{e}_i$.

$(\mathcal{P}(\boldsymbol{x}') \leftrightarrow \mathcal{V})(\boldsymbol{A}, \boldsymbol{A}')$: Le prouveur ayant une clé secrète $\boldsymbol{x}' = \boldsymbol{x}_i\| - \boldsymbol{e}_i$ procède de la manière suivante :

1. $\mathcal{P}$ choisit $\pi_m \leftarrow S_m$, $\pi_\ell \leftarrow S_\ell$ et $\boldsymbol{u} \leftarrow \mathbb{Z}_q^{m+\ell}$. Soit $\pi = \pi_m \times \pi_\ell$, $\mathcal{P}$ envoie c_0, c_1 et c_3 à $\mathcal{V}$ tels que
   — $c_1 = \mathsf{COM}(\pi_m\|\pi_\ell\|\boldsymbol{A}'\boldsymbol{u})$
   — $c_2 = \mathsf{COM}(\pi(\boldsymbol{u}))$
   — $c_3 = \mathsf{COM}(\pi(\boldsymbol{x}' + \boldsymbol{u}))$
2. $\mathcal{V}$ choisit un challenge $ch \in \{1,2,3\}$ et il l'envoie à $\mathcal{P}$
3. — Si $ch = 1$, $\mathcal{P}$ envoie $\boldsymbol{s} = \pi(\boldsymbol{x}')$ et $\boldsymbol{w} = \pi(\boldsymbol{u})$.
   — Si $ch = 2$, $\mathcal{P}$ envoie $\phi_m = \pi_m, \phi_\ell = \pi_\ell$ et $\boldsymbol{v} = \boldsymbol{x}' + \boldsymbol{u}$.
   — Si $ch = 3$, $\mathcal{P}$ envoie $\varphi_m = \pi_m$, $\varphi_\ell = \pi_\ell$ et $\boldsymbol{z} = \boldsymbol{u}$.

   . Vérification :
   — Si $ch = 1$, $\mathcal{V}$ vérifie si $c_2 = \mathsf{COM}(\boldsymbol{w})$, $c_3 = \mathsf{COM}(\boldsymbol{s} + \boldsymbol{w})$, et $\boldsymbol{s}$ est de la forme $\boldsymbol{s}_m\| - \boldsymbol{e}_j$ pour $j \in [\ell]$, et $\boldsymbol{s}_m \in \{0,1\}^m$ tel que $w_H(\boldsymbol{s}_m) = m/2$.
   — Si $ch = 2$, $\mathcal{V}$ vérifie si $c_1 = \mathsf{COM}(\phi_m\|\phi_\ell\|\boldsymbol{A}'\boldsymbol{v} - \boldsymbol{y})$ et $c_3 = \mathsf{COM}((\phi_m \times \phi_\ell)(\boldsymbol{v}))$.
   — Si $ch = 3$, $\mathcal{V}$ vérifie si $c_1 = \mathsf{COM}(\varphi_m\|\varphi_\ell\|\boldsymbol{A}'\boldsymbol{z})$ et $c_2 = \mathsf{COM}((\varphi_m \times \varphi_\ell)(\boldsymbol{z}))$.

   $\mathcal{V}$ retourne $\mathsf{desc} = 1$ si les tests sont valides, sinon il retourne $\mathsf{desc} = 0$.

Sécurité : Nous rappelons brièvement les idées des preuves de sécurité du schéma :

— Usurpation d'identité : La stratégie proposé par Kawachi et al. pour avoir la sécurité contre l'usurpation d'identité avec des attaques parallèles à cercle choisi, est très semblable à celle utilisée pour la preuve du schéma d'identification KTX.
Au début de la réduction, une matrice $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$ est choisie de manière uniformément aléatoire et donnée au challengeur $\mathcal{C}$. Ce dernier va essayer de résoudre l'instance $\mathsf{SIS}_{n,q,m,\sqrt{m}}$ reliée à cette matrice en utilisant l'attaquant $\mathcal{I}$, autrement dit il faut qu'il trouve un vecteur $\boldsymbol{s} \in \mathbb{Z}^m \setminus \{0\}$ tel que $\boldsymbol{As} = 0 \bmod q$ et $\|\boldsymbol{s}\|_2 \leq \sqrt{m}$. Au début du jeu $\mathcal{C}$ utilise l'algorithme de génération des clés pour obtenir des paires de clés valides $\{(\boldsymbol{x}_1, \boldsymbol{y}_1), \cdots, (\boldsymbol{x}_\ell, \boldsymbol{y}_\ell)\}$. Ainsi, pendant la phase des requêtes, le challengeur peut répondre à toutes les requêtes de l'adversaire. En utilisant les réponses de $\mathcal{I}$ pendant la phase de challenge, le challengeur obtient une collision sur la fonction d'engagement COM ou bien un vecteur $\boldsymbol{x}'$ tel qu'il existe $i \in [\ell]$ avec $\boldsymbol{x}' \neq \boldsymbol{x}_i$, $\boldsymbol{Ax}' = \boldsymbol{Ax}_i = \boldsymbol{y}_i \bmod q$, $\boldsymbol{x}' \in \{0,1\}^m$ et $w_H(\boldsymbol{x}') = m/2$. Finalement le challengeur retourne le vecteur $\boldsymbol{s} = \boldsymbol{x}' - \boldsymbol{x}_i$ comme solution à l'instance $\mathsf{SIS}_{n,q,m,\sqrt{m}}$.

— Anonymat : Le schéma satisfait la propriété de témoin indistinguable par conséquent ceci implique l'anonymat suivant la Définition 3.3.

3.1.3 Notre schéma d'identification de cercle

Dans cette section, nous décrivons un schéma d'identification de cercle qui est une variante plus efficace du schéma d'identification de cercle proposé par Kawachi et al [KTX08] (voir la section 3.1.2).

3.1.3.1 Aperçu

Soient n le paramètre de sécurité et q, m deux entiers tels que $q = \mathsf{poly}(n)$ et $m \geq 10n \log q$. Le protocole est paramétré par une matrice $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$ choisie de manière uniformément aléatoire et rendue publique. Chaque prouveur i dans le cercle admet une paire de clés : une clé secrète qui consiste en un vecteur $\boldsymbol{x}_i \in \{0,1\}^m$ tel que $w_H(\boldsymbol{x}_i) = m/2$ et une clé publique qui correspond à un vecteur $\boldsymbol{y}_i \in \mathbb{Z}_q^n$ tel que $\boldsymbol{y}_i = \boldsymbol{Ax}_i \bmod q$.

Supposons que nous avons un cercle $R = (\boldsymbol{y}_1, \cdots, \boldsymbol{y}_\ell)$ de taille ℓ où chaque vecteur $\boldsymbol{y}_i$ correspond à la clé publique d'un utilisateur i. Soit $\boldsymbol{M}$ la matrice ayant les vecteurs $\boldsymbol{y}_i$ comme colonnes. Pour $i \in [\ell]$, soit $\boldsymbol{e}_i$ le vecteur dans $\{0,1\}^\ell$ ayant 1

à la i-ème coordonnée et 0 autre part.

$$\boldsymbol{M} = \begin{pmatrix} | & | & \cdots & | \\ \boldsymbol{y}_1 & \boldsymbol{y}_2 & \cdots & \boldsymbol{y}_N \\ | & | & \cdots & | \end{pmatrix}, \boldsymbol{e}_i = \underbrace{(0, \cdots, 1, \cdots, 0)}_{\text{1 à la i-ème position}} \in \{0,1\}^\ell.$$

Soit $\mathcal{P}$ un prouveur ayant une paire de clés $(pk = \boldsymbol{y}_i, sk = \boldsymbol{x}_i)$ tel que $\boldsymbol{y}_i \in R$. Nous supposons que $\mathcal{P}$ veut s'identifier en tant qu'un membre du cercle R auprès d'un vérificateur $\mathcal{V}$. Pour cela $\mathcal{P}$ donne une preuve de connaissance sans divulgation de deux vecteurs $(\boldsymbol{x}_i, \boldsymbol{e}_i)$ telle que

1. $\boldsymbol{A}\boldsymbol{x}_i - \boldsymbol{M}\boldsymbol{e}_i = 0 \bmod q$.
2. $(\boldsymbol{x}_i, \boldsymbol{e}_i) \in \{0,1\}^m \times \{0,1\}^\ell$, $w_H(\boldsymbol{x}_i) = m/2$ et $w_H(\boldsymbol{e}_i) = 1$.

Nous remarquons que n'importe quel membre du cercle peut convaincre le vérificateur de cette façon puisque pour tout $j \in [\ell]$, nous avons $\boldsymbol{A}\boldsymbol{x}_j - \boldsymbol{M}\boldsymbol{e}_j = 0 \bmod q$.

3.1.3.2 Description

$\boldsymbol{A} \leftarrow \mathsf{Ring{-}id{-}init}(1^n)$: Étant donné un paramètre de sécurité n. Choisir une matrice $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$ d'une façon uniformément aléatoire.

$(\boldsymbol{y}_i, \boldsymbol{x}_i) \leftarrow \mathsf{Ring{-}id{-}keygen}(\boldsymbol{A}, i)$: Étant donnés une matrice $\boldsymbol{A}$ et un entier i correspondant à l'indice de l'utilisateur dans le cercle, choisir une paire de clés en suivant la procédure suivante :
— Soit $\boldsymbol{x}_i \leftarrow \{0,1\}^m$, tel que $w_H(\boldsymbol{x}_i) = m/2$.
— Soit $\boldsymbol{y}_i = \boldsymbol{A}\boldsymbol{x}_i \bmod q$.

$\boldsymbol{M} \leftarrow \mathsf{Ring{-}id{-}ccp}(\boldsymbol{A}, R = (\boldsymbol{y}_1, \cdots, \boldsymbol{y}_\ell))$: La clé publique du cercle R est une matrice $\boldsymbol{M} \in \mathbb{Z}_q^{n \times \ell}$, telle que $\boldsymbol{M} = \begin{pmatrix} | & | & \cdots & | \\ \boldsymbol{y}_1 & \boldsymbol{y}_2 & \cdots & \boldsymbol{y}_\ell \\ | & | & \cdots & | \end{pmatrix}$.

$(\boldsymbol{x}_i, \boldsymbol{e}_i) \leftarrow \mathsf{Ring{-}id{-}ccs}(\boldsymbol{A}, R = (\boldsymbol{y}_1, \cdots, \boldsymbol{y}_\ell), \boldsymbol{x}_i)$: La clé secrète du cercle est construite en suivant la procédure suivante :
— Soit $\boldsymbol{e}_i \in \{0,1\}^m$ le vecteur ayant 1 comme i-ème coordonnée et $w_H(\boldsymbol{e}_i) = 1$.

$(\mathcal{P}(\boldsymbol{x}_i, \boldsymbol{e}_i) \leftrightarrow \mathcal{V})(\boldsymbol{A}, \boldsymbol{M})$: Le prouveur ayant une clé secrète $(\boldsymbol{x}_i, \boldsymbol{e}_i)$ procède de la manière suivante :

1. $\mathcal{P}$ choisit les permutation $\Sigma \leftarrow S_\ell$, $\pi \leftarrow S_m$. Soient $\boldsymbol{u} \leftarrow \mathbb{Z}_q^m$ et $\boldsymbol{u}' \leftarrow \mathbb{Z}_q^\ell$. $\mathcal{P}$ envoie c_0 et c_1 au vérificateur $\mathcal{V}$ tels que
   — $c_0 = \mathsf{COM}(\pi || \Sigma || \boldsymbol{A}\boldsymbol{u} - \boldsymbol{M}\boldsymbol{u}')$
   — $c_1 = \mathsf{COM}(\pi(\boldsymbol{u}) || \Sigma(\boldsymbol{u}') || \pi(\boldsymbol{x}_i) || \Sigma(\boldsymbol{e}_i))$
2. $\mathcal{V}$ envoie $\alpha \leftarrow \mathbb{Z}_q$ à $\mathcal{P}$.

3. $\mathcal{P}$ envoie $\boldsymbol{g}$ et $\boldsymbol{g}'$ à $\mathcal{V}$, tels que

$$\boldsymbol{g} = \pi(\boldsymbol{u} + \alpha \boldsymbol{x}_i) \text{ et } \boldsymbol{g}' = \Sigma(\boldsymbol{u}' + \alpha \boldsymbol{e}_i).$$

4. $\mathcal{V}$ envoie $b \leftarrow \{0, 1\}$ à $\mathcal{P}$.
5. Si $b = 0$ alors
   $\mathcal{P}$ envoie $\phi = \Sigma$ et $\psi = \pi$, à $\mathcal{V}$ qui vérifie si

$$c_0 = \mathsf{COM}(\psi \| \phi \| \boldsymbol{A}\psi^{-1}(\boldsymbol{g}) - \boldsymbol{M}\phi^{-1}(\boldsymbol{g}'))$$

Si $b = 1$ alors
$\mathcal{P}$ envoie $\boldsymbol{w} = \Sigma(\boldsymbol{e}_i), \boldsymbol{v} = \pi(\boldsymbol{x}_i)$ à $\mathcal{V}$ qui vérifie si

$$c_1 = \mathsf{COM}((\boldsymbol{g} - \alpha \boldsymbol{v}) \| (\boldsymbol{g}' - \alpha \boldsymbol{w}) \| \boldsymbol{v} \| \boldsymbol{w}),$$

$$\boldsymbol{v} \in \{0, 1\}^m, \boldsymbol{w} \in \{0, 1\}^\ell \text{ , } w_H(\boldsymbol{v}) = m/2 \text{ et } w_H(\boldsymbol{w}) = 1$$

$\mathcal{V}$ retourne desc $= 1$ si les tests sont valides, sinon il retourne desc $= 0$.

3.1.3.3 Sécurité

Théorème 3.1. *Si le schéma d'engagement satisfait la propriété d'indistinguabilité, alors notre schéma d'identification de cercle est statistiquement sans divulgation.*

Démonstration. Pour démontrer le théorème nous construisons un simulateur $\mathcal{S}$ qui prend comme entrées les matrices $\boldsymbol{A}$ et $\boldsymbol{M}$ et qui a accès à un vérificateur malhonnête $\tilde{\mathcal{V}}$ qui fournit des challenges $(\alpha \in \mathbb{Z}_q, b \in \{0, 1\})$. Ce simulateur retourne une distribution statistiquement proche de la vue d'un vérificateur honnête $\mathcal{V}$ lorsque il interagit avec un prouveur honnête $\mathcal{P}$.

Le simulateur doit deviner le challenge b, avant d'interagir avec $\tilde{\mathcal{V}}$, nous supposons que $\mathcal{S}$ devine correctement le challenge :

— Si $b = 0$, le simulateur choisit $\Sigma, \pi, \boldsymbol{u}$ et $\boldsymbol{u}'$ comme dans le protocole. Puis il résout l'équation $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}_j \bmod q$ pour $\boldsymbol{y}_j$ choisi au hasard parmi les colonnes de la matrice $\boldsymbol{M}$. Le simulateur construit le vecteur $\boldsymbol{e}_j$ qui correspond à $\boldsymbol{y}_j$. Puis il calcule c_0, c_1 comme dans le protocole. Après avoir reçu α de la part du vérificateur $\tilde{\mathcal{V}}$, le simulateur calcule $\boldsymbol{g} = \pi(\boldsymbol{u} + \alpha \boldsymbol{x})$ et $\boldsymbol{g}' = \Sigma(\boldsymbol{u}' + \alpha \boldsymbol{e}_j)$. Les valeurs de $\boldsymbol{g}$ et $\boldsymbol{g}'$ ont une distribution uniforme puisque les vecteurs $\boldsymbol{u}$ et $\boldsymbol{u}'$ ont été choisi de manière uniformément aléatoire. Ainsi, lorsque le simulateur envoie Σ et σ, nous obtenons une vérification valide pour $b = 0$. D'autre part, grâce à la propriété d'indistinguabilité du schéma d'engagement, nous avons que la distribution de (c_0, c_1) est indistinguable de la distribution d'un couple qui aurait pu être générée par un prouveur honnête $\mathcal{P}$.

— Si $b = 1$, le simulateur choisit $\Sigma, \sigma, \boldsymbol{u}$ et $\boldsymbol{u}'$ comme dans le protocole. Il choisit aussi $\boldsymbol{x} \in \{0,1\}^m$ tel que $w_H(\boldsymbol{x}) = m/2$ et $\boldsymbol{e} \in \{0,1\}^\ell$ tel que $w_H(\boldsymbol{e}) = 1$. Ensuite il calcule les deux challenges c_0 et c_1, après la réception de α de la part de $\tilde{\mathcal{V}}$, il calcule $\boldsymbol{g}$ et $\boldsymbol{g}'$ comme dans le protocole. Lorsque le simulateur révèle $\Sigma(\boldsymbol{e})$ et $\pi(\boldsymbol{x})$, nous obtenons une vérification valide pour $b = 1$. Pour les mêmes arguments avancés dan le cas $(b = 0)$, nous avons que la distribution des engagements (c_0, c_1) est indistinguable de la distribution des engagements issus d'un protocole joué de manière conforme.

La probabilité que le simulateur produise une simulation correcte est de $1/2$. Lorsque le simulateur devine correctement le challenge b du vérificateur, il produit une transcription statistiquement proche d'une l'interaction réelle entre un prouveur $\mathcal{P}$ et un vérificateur $\mathcal{V}$. Le simulateur enregistre les valeurs de c_0, c_1, α, $\boldsymbol{g}, \boldsymbol{g}'$ et la réponse au challenge b lorsqu'il devine correctement la valeur de b. Sinon puisque le simulateur a accès à un vérificateur malhonnête, il peut le relancer de nouveau. □

Corollaire 3.1. *Si le schéma d'engagement satisfait la propriété d'indistinguabilité, alors notre schéma d'identification de cercle vérifie l'anonymat contre la révélation totale des clés.*

Démonstration. En combinant le Théorème 3.1 et la Proposition 1.4, nous obtenons que notre schéma est à témoin indistinguable.

Supposons que l'attaquant $\mathcal{A}$ envoie $(R, (pk_{i_0}, sk_{i_0}), (pk_{i_1}, sk_{i_1}))$ comme challenge au challengeur $\mathcal{C}$, avec $R = \{pk_i\}_{i\in[\ell]}$, $\{pk_{i_0}, pk_{i_1}\} \in R$ et $(pk_{i_0}, sk_{i_0}), (pk_{i_1}, sk_{i_1})$. Nous montrons que les transcriptions dans les deux cas, c'est-à-dire lorsque $\mathcal{C}$ exécute le protocole avec sk_{i_0} ou bien sk_{i_1}, sont statistiquement indistinguables.

En effet, puisque notre schéma est à témoin indistinguable alors nous avons que :

$$\mathsf{Vue}_{\mathcal{A}}\left[(\mathcal{C}(sk_{i_0}) \leftrightarrow \mathcal{A}(sk_{i_0} \| sk_{i_1}))(R)\right] \approx_S \mathsf{Vue}_{\mathcal{A}}\left[(\mathcal{C}(sk_{i_1}) \leftrightarrow \mathcal{A}(sk_{i_0} \| sk_{i_1}))(R)\right].$$

□

Théorème 3.2. *S'il existe un attaquant $\mathcal{I}$ qui réussit dans le jeu de la Définition 3.2 avec une probabilité supérieure à $\frac{q+1}{2q} + \varepsilon$, avec $\varepsilon = \varepsilon(n)$ non négligeable. Alors il existe un algorithme probabiliste polynomial $\mathcal{A}$ qui résout une instance aléatoire du problème $\mathsf{SIS}_{n,q,m,\sqrt{m}}$.*

Démonstration. La preuve est divisée en deux parties. Dans la première parties nous montrons qu'un prouveur malhonnête (qui ne connait aucune des clés secrètes) est capable d'usurper l'identité d'un utilisateur arbitraire du cercle avec une probabilité au plus égale à $1/2$ (plus précisément nous montrons que cette probabilité est égale à $(q+1)/2q \simeq 1/2$). Dans la deuxième partie nous montrons

qu'un attaquant $\mathcal{I}$ capable de gagner le jeu de la Définition 3.2, avec une probabilité supérieure à $\frac{q+1}{2q} + \varepsilon$, pourrait être utiliser pour construire un algorithme $\mathcal{A}$ qui résout une instance aléatoire du problème $\mathsf{SIS}_{n,q,m,\sqrt{m}}$.

Partie 1. On considère un cercle R de taille ℓ défini par les deux matrices $(\boldsymbol{A}, \boldsymbol{M})$ tel que $\boldsymbol{A}$ est la matrice publique et $\boldsymbol{M}$ est la matrice de clés publiques $\boldsymbol{y}_i$. Nous savons que le nombre total des requêtes possibles à envoyer à un pouveur est égal à $2q$. En effet ce nombre correspond à toutes les combinaisons des challenges $\alpha \in \mathbb{Z}_q$ et $b \in \{0,1\}$. Maintenant supposons que le prouveur agisse suivant deux stratégies notées ST_0 et ST_1 pour répondre aux challenges envoyés par le vérificateur.

— ST_0 : Dans cette stratégie, nous supposons que le prouveur répond aux challenges qui correspondent à $b = 0$. Pour cela il choisit au hasard $\boldsymbol{y}_i$ parmi les colonnes de la matrice $\boldsymbol{M}$, il calcule $\boldsymbol{x}'_i$ quelconque tel que $\boldsymbol{A}\boldsymbol{x}'_i = \boldsymbol{y}_i \bmod q$. Ensuite il calcule c_0 comme dans le schéma et il affecte à c_1 une valeur choisie de manière uniformément aléatoire. Par conséquent le prouveur peut répondre correctement au challenge $b = 0$ indépendamment de la valeur de α.

— ST_1 : Dans cette stratégie, nous supposons que le prouveur répond aux challenges qui correspondent à $b = 1$. Pour cela il choisit $\boldsymbol{x}_i \in \{0,1\}^m$ tel que $w_H(\boldsymbol{x}_i) = m/2$ et $\boldsymbol{e}_i \in \{0,1\}^\ell$ qui a 1 à la i-ème coordonnée et 0 autre part. Il choisit aléatoirement $\boldsymbol{u}, \boldsymbol{u}', \pi$ et Σ comme dans le protocole. En utilisant ces valeurs, il calcule c_1 suivant le protocole et il affecte à c_0 une valeur choisie de manière uniformément aléatoire. Par conséquent le prouveur peut répondre correctement au challenge $b = 1$ indépendamment de la valeur de α.

L'attaquant peut améliorer ces stratégies, par exemple pour la stratégie ST_0 il va pouvoir répondre non seulement au challenge $b = 0$ pour tout les $\alpha \in \mathbb{Z}_q$ mais aussi au challenge $b = 1$ pour un α_0 choisi à l'avance. Dans la stratégie ST_0 au lieu de générer c_1 d'une manière aléatoire, il va procéder de la manière suivante : il choisit $\boldsymbol{s}_1 \in \{0,1\}^m$ tel que $w_H(\boldsymbol{s}_1) = m/2$, $\boldsymbol{s}_2 \in \{0,1\}^\ell$ tel que $w_H(\boldsymbol{s}_2) = 1$ et $\boldsymbol{u}$, $\boldsymbol{u}'$, π et Σ comme dans le protocole, ensuite il calcule

$$\boldsymbol{g} = \pi(\boldsymbol{u} - \alpha \boldsymbol{x}_i) \qquad \boldsymbol{g}' = \Sigma(\boldsymbol{u}' - \alpha \boldsymbol{e}_i).$$

Avec $\boldsymbol{x}_i$ et $\boldsymbol{e}_i$ sont choisis comme il est décrit dans la stratégie ST_0 ci-dessus. Ensuite il calcule c_1 de la manière suivante

$$c_1 = \mathsf{COM}(\boldsymbol{g} - \alpha_0 \boldsymbol{v} \| \boldsymbol{g}' - \alpha_0 \boldsymbol{w} \| \boldsymbol{v} \| \boldsymbol{w}),$$

avec $\boldsymbol{v} = \pi(\boldsymbol{s}_1)$ et $\boldsymbol{w} = \Sigma(\boldsymbol{s}_2)$. Ainsi le prouveur peut répondre au challenge $b = 0$ indépendamment du challenge α choisi par le vérificateur et il peut aussi répondre pour le challenge $b = 1$ si $\alpha = \alpha_0$.

Dans la stratégie ST_1 au lieu de générer c_0 d'une manière aléatoire, il va procéder de la manière suivante : il choisit $\boldsymbol{x}\{0,1\}^m$ tel que $w_H(\boldsymbol{x}) = m/2$, $\boldsymbol{e} \in \{0,1\}^\ell$

tel que $w_H(\boldsymbol{e}_i) = 1$ et $\boldsymbol{u}$, $\boldsymbol{u}'$, π et Σ comme dans le protocole, ensuite il calcule

$$c_0 = \mathsf{COM}(\pi \| \Sigma \| \boldsymbol{A}\boldsymbol{u} - \boldsymbol{M}\boldsymbol{u}' - \alpha_0(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{M}\boldsymbol{e})) \qquad c_1 = \mathsf{COM}(\pi(\boldsymbol{u}) \| \Sigma(\boldsymbol{u}') \| \pi(\boldsymbol{x}) \| \Sigma(\boldsymbol{e}))$$

$$\boldsymbol{g} = \pi(\boldsymbol{u} - \alpha\boldsymbol{x}) \qquad \boldsymbol{g}' = \Sigma(\boldsymbol{u}' - \alpha\boldsymbol{e}).$$

Ainsi le prouveur peut répondre au challenge $b = 1$ indépendamment du challenge α choisi par le vérificateur et il peut aussi répondre pour $b = 0$ si $\alpha = \alpha_0$.

Par conséquent, dans les deux stratégies, le prouveur peut répondre à $q + 1$ challenges parmi les $2q$ challenges possibles. Ceci implique que la probabilité du succès du prouveur pour un tour du protocole est égale à $\frac{q+1}{2q}$.

Partie 2. Au début l'algorithme $\mathcal{A}$ reçoit comme challenge SIS une matrice $\boldsymbol{A}$ choisie de manière uniformément aléatoire dans $\mathbb{Z}_q^{n \times m}$. Ensuite, il initialise l'attaquant $\mathcal{I}$ en lui donnant la matrice $\boldsymbol{A}$. L'attaquant $\mathcal{I} = (\tilde{\mathcal{P}}, \tilde{\mathcal{V}})$, utilise son oracle $\tilde{\mathcal{V}}$ pour jouer le rôle d'un vérificateur malhonnête pendant la phase d'apprentissage. A la fin du jeu, il utilise l'oracle $\tilde{\mathcal{P}}$ (phase du challenge) pour usurper l'identité d'un utilisateur qui appartient à un cercle de son choix.

Pendant le jeu, $\mathcal{A}$ simule les oracles Init, Corrupt, et Prouv de la manière suivante. Pour oracle Init, lorsque $\mathcal{I}$ envoie une requête (i), $\mathcal{A}$ choisit $\boldsymbol{x}_i \in \{0,1\}^m$ tel que $w_H(\boldsymbol{x}_i) = m/2$ ensuite il calcule $\boldsymbol{y}_i = \boldsymbol{A}\boldsymbol{x}_i \bmod q$ et il envoie $\boldsymbol{y}_i$ à $\mathcal{I}$. $\mathcal{A}$ peut répondre aux requêtes pour les oracles Corrupt et Prouv puisqu'il connait toutes les clés secrètes.

Dans la phase du challenge, $\mathcal{I}$ envoie comme challenge un cercle défini par l'ensemble des clés publiques $R_c = \{\boldsymbol{y}_1, \cdots, \boldsymbol{y}_c\}$, nous notons la matrice correspondante par $\boldsymbol{M}_c$ (i.e., $\boldsymbol{M}_c = [\boldsymbol{y}_1 \| \cdots \| \boldsymbol{y}_c]$). Ensuite il essaye de convaincre $\mathcal{A}$ qu'il connait une clé secrète qui correspond à l'un des $\boldsymbol{y}_i \in R_c$.

Pendant cette phase $\mathcal{A}$ joue le rôle d'un vérificateur. Au début $\mathcal{A}$ reçoit les deux engagements c_0 et c_1 de la part de $\tilde{\mathcal{P}}$. Ensuite $\mathcal{A}$ interroge $\tilde{\mathcal{P}}$ avec tous les challenges possibles (α, b), cela est possible parce que le nombre total de ses requêtes est $2q$ qui est polynomial en le paramètre de sécurité n. $\mathcal{A}$ enregistre les réponses de $\tilde{\mathcal{P}}$ dans le tableau suivant :

	$b = 0$	$b = 1$
$\alpha = 0$	$\mathsf{desc}_{(0,0)}$	$\mathsf{desc}_{(0,1)}$
$\alpha = 1$	$\mathsf{desc}_{(1,0)}$	$\mathsf{desc}_{(1,1)}$
$\vdots$	$\vdots$	$\vdots$
$\alpha = q - 1$	$\mathsf{desc}_{(q-1,0)}$	$\mathsf{desc}_{(q-1,1)}$

Pour $(i, j) \in \mathbb{Z}_q \times \{0,1\}$, $\mathsf{desc}_{(i,j)}$ appartient à $\{0,1\}$ et désigne la réponse finale du vérificateur.

Supposons qu'il existe deux lignes α_1 et α_2 dans le tableau telles que pour chaque ligne on a $\mathsf{desc} = 1$ dans les deux colonnes (colonnes $b = 0$ et $b = 1$). Soient

$((\boldsymbol{g}_1, \boldsymbol{g}'_1), (\psi_1, \phi_1))$ et $((\boldsymbol{g}_2, \boldsymbol{g}'_2), (\psi_2, \phi_2))$ les résultats des challenges $(\alpha = \alpha_1, b = 0)$ et $(\alpha = \alpha_2, b = 0)$ respectivement. Soient $((\boldsymbol{g}_1, \boldsymbol{g}'_1), (\boldsymbol{v}_1, \boldsymbol{w}_1))$ et $((\boldsymbol{g}_2, \boldsymbol{g}'_2), (\boldsymbol{v}_2, \boldsymbol{w}_2))$ les résultats des challenges $(\alpha = \alpha_1, b = 1)$ et $(\alpha = \alpha_2, b = 1)$ respectivement.

Maintenant nous analysons les réponses de $\tilde{\mathcal{P}}$. En utilisant le fait que le schéma d'engagement satisfait la propriété d'engagement, nous déduisons que :

$$\psi_1 \|\phi_1\| \boldsymbol{A}\psi_1^{-1}(\boldsymbol{g}_1) - \boldsymbol{M}_c\phi_1^{-1}(\boldsymbol{g}'_1) = \psi_2 \|\phi_2\| \boldsymbol{A}\psi_2^{-1}(\boldsymbol{g}_2) - \boldsymbol{M}_c\phi_2^{-1}(\boldsymbol{g}'_2) \tag{3.1}$$

$$\boldsymbol{g}_1 - \alpha_1 \boldsymbol{v}_1 \|\boldsymbol{g}'_1 - \alpha_1 \boldsymbol{w}_1 \|\boldsymbol{v}_1\| \boldsymbol{w}_1 = \boldsymbol{g}_2 - \alpha_2 \boldsymbol{v}_2 \|\boldsymbol{g}'_2 - \alpha_2 \boldsymbol{w}_2 \|\boldsymbol{v}_2\| \boldsymbol{w}_2 \tag{3.2}$$

D'après les équations (3.1) et (3.2), nous avons que $\psi_1 = \psi_2, \phi_1 = \phi_2, \boldsymbol{v}_1 = \boldsymbol{v}_2$ et $\boldsymbol{w}_1 = \boldsymbol{w}_2$. Nous avons aussi que $\boldsymbol{v}_1 \in \{0,1\}^m$ tel que $w_H(\boldsymbol{v}_1) = m/2$ et $\boldsymbol{w}_1 \in \{0,1\}^c$ tel que $w_H(\boldsymbol{w}_1) = 1$. En réarrangeant les termes des équations (3.1) et (3.2) nous obtenons,

$$\boldsymbol{A}\psi_1^{-1}(\boldsymbol{g}_1 - \boldsymbol{g}_2) - \boldsymbol{M}_c\phi^{-1}(\boldsymbol{g}'_1 - \boldsymbol{g}'_2) = 0 \bmod q. \tag{3.3}$$

$$\boldsymbol{g}_1 - \boldsymbol{g}_2 = \boldsymbol{v}_1(\alpha_1 - \alpha_2) \bmod q. \tag{3.4}$$

$$\boldsymbol{g}'_1 - \boldsymbol{g}'_2 = \boldsymbol{w}_1(\alpha_1 - \alpha_2) \bmod q. \tag{3.5}$$

Il est clair que $\boldsymbol{g}_1 - \boldsymbol{g}_2 \neq 0$ et $\boldsymbol{g}'_1 - \boldsymbol{g}'_2 \neq 0$, et d'après ces équations nous obtenons que $\boldsymbol{A}\psi_1^{-1}(\boldsymbol{v}_1) - \boldsymbol{M}_c\phi_1^{-1}(\boldsymbol{w}_1) = 0 \bmod q$. Donc $\boldsymbol{x}'_k = \psi_1^{-1}(\boldsymbol{v}_1)$ est un secret valide pour $\boldsymbol{y}_k = \boldsymbol{M}_c\phi_1^{-1}(\boldsymbol{w}_1)$ avec $k \in [c]$ (autrement dit $\boldsymbol{y}_k = \boldsymbol{A}\boldsymbol{x}'_k \bmod q$).

Maintenant nous allons montrer que la clé extraite est différente de la clé généré par $\mathcal{A}$ avec une probabilité au moins égale à $1/2$. Pour montrer cela nous rappelons que d'après le Lemme 1.6 il existe un autre secret valide pour $\boldsymbol{y}_k$ avec une probabilité proche de un. Nous savons aussi que le protocole est à témoins indistinguable, ce qui implique que la clé $\boldsymbol{x}'_k$ extraite grâce à l'attaquant $\tilde{\mathcal{P}}$ est indépendante de celle qui pourrait être utilisé par $\mathcal{A}$ durant la phase d'apprentissage.

Par conséquent, $\mathcal{A}$ obtient $\boldsymbol{s} = \boldsymbol{x}_k - \boldsymbol{x}'_k \in \{-1, 0, 1\}^m$, tel que $\boldsymbol{A}\boldsymbol{s} = 0 \bmod q$ et $\|\boldsymbol{s}\|_2 \leq \sqrt{m}$. $\mathcal{A}$ retourne le vecteur $\boldsymbol{s}$ comme solution au problème $\mathsf{SIS}_{n,q,m,\sqrt{m}}$ défini par la matrice $\boldsymbol{A}$.

Maintenant nous montrons que le tableau ci-dessus contient deux lignes qui correspondent à (α_1, α_2). D'après la première partie de la preuve, nous savons que $\tilde{\mathcal{P}}$ peut répondre correctement à $q+1$ challenges parmi $2q$. Donc si $\tilde{\mathcal{P}}$ répond à $q+2$ challenges, alors en utilisant le principe des tiroirs nous avons qu'au pire cas il y a q réponses égales à 1 dans la colonne $b = 0$ et au moins deux réponses égales à 1 dans la colonnes de $b = 1$.

□

D'après la Proposition 1.4, nous avons que la propriété de témoin indistinguable est préservée dans le cas d'exécution parallèle. Par conséquent le protocole peut être exécuté $t = \omega(\log n)$ fois en parallèle pour obtenir une faible probabilité de fraude.

3.1.3.4 Coût et complexité

Dans cette section nous examinons la complexité de notre schéma d'identification de cercle et la comparons au schéma de Kawachi et al.

Tout d'abord nous rappelons que la probabilité de triche (coté prouveur) pour chaque protocole a un impact direct sur le coût total de communication. Pour avoir une probabilité de triche au dessous de $\frac{1}{2^{16}}$, il faut que :

$$(\frac{2}{3})^{\delta} \leq \frac{1}{2^{16}} \quad \text{et} \quad (\frac{q+1}{2q})^{\delta} \leq \frac{1}{2^{16}}$$

avec δ le nombre d'instances nécessaire. On en déduit qu'il faut 28 instances pour le schéma de Kawachi et al., tandis qu'il faut 17 instances pour notre protocole.

Pour calculer le coût de communication des deux protocoles, nous supposons que le prouveur et le vérificateur partagent un générateur aléatoire. Par conséquent, lorsque un vecteur aléatoire doit être envoyé par le prouveur au vérificateur, le premier envoie seulement la graine (*seed*) qui permet de l'obtenir à partir du générateur aléatoire. Dans les schémas que nous souhaitons comparer, ceci concerne les vecteurs qui représentent les permutations et les vecteurs qui permettent de vérifier un engagement (ces derniers, d'après la section 1.5.1.4, sont dans $\{0,1\}^{m/2}$). Nous notons le nombre de bits dans une graine par $|seed|$.

Dans la suite, nous calculons le coût de communication dans le pire cas, c'est-à-dire dans le cas où le challenge du vérificateur est égal à 1 dans les deux protocoles. C'est l'échange le plus coûteux pour les deux protocoles.

— Pour le protocole de Kawachi et al., la transcription est de la forme suivante :

$$((c_1, c_2, c_3); ch; (\boldsymbol{s}, \boldsymbol{w}, \boldsymbol{r}_2, \boldsymbol{r}_3))$$

avec pour $i \in [3]$, $c_i \in \mathbb{Z}_q^n$, $\boldsymbol{s} \in \{-1, 0, 1\}^{m+\ell}$, $\boldsymbol{w} \in \mathbb{Z}_q^{m+\ell}$ et $\boldsymbol{r}_2, \boldsymbol{r}_3$ deux vecteurs aléatoires nécessaires pour la vérification de C_2 et C_3 respectivement. Par conséquent le coût de cette transcription en bits est

$$3 \times n \log q + (m+\ell) \log q + (m+\ell) + 2 \times |seed|.$$

— Pour notre protocole, la transcription est de la forme suivante :

$$((c_0, c_1); \alpha; (\boldsymbol{g}, \boldsymbol{g'}); b; (\boldsymbol{w}, \boldsymbol{v}, \boldsymbol{r}_1)$$

avec $c_0, c_1 \in \mathbb{Z}_q^n$, $\alpha \in \mathbb{Z}_q$, $(\boldsymbol{g}, \boldsymbol{g'}) \in \mathbb{Z}_q^{\ell} \times \mathbb{Z}_q^m$, $(\boldsymbol{w}, \boldsymbol{v}) \in \{0,1\}^{\ell} \times \{0,1\}^m$ et $\boldsymbol{r}_1$ un vecteur aléatoire nécessaire pour la vérification de c_2. Par conséquent le coût de cette transcription en bits est

$$2 \times n \log q + \log q + (m+\ell) \log q + (m+\ell) + |seed|.$$

Notre protocole peut être effectué avec 17 instances au lieu de 28 pour le schéma de Kawachi et al. et nous avons approximativement $(n-1) \log q$ bits en moins pour chaque instance.

3.2 Schémas d'identification de cercle à seuil

Dans cette section, nous nous intéressons à la généralisation de notre schéma d'identification de cercle présenté dans la section 3.1.3. Nous construisons un protocole d'identification interactif qui permet à t parmi N utilisateurs de collaborer ensemble pour s'identifier auprès d'un vérificateur d'une manière anonyme. Le cas $t = 1$ correspond, bien évidemment, à un schéma d'identification de cercle.

3.2.1 Définition

Soient t, N deux entiers tels que $t < N$ et U, P deux ensembles tels que $U = [N]$, $P \subset U$ de taille égale à t. On considère qu'il existe une bijection entre les éléments de U et les utilisateurs dans le protocole (chaque entier i correspond à un utilisateur i identifié par une paire de clés (pk_i, sk_i)). Pour simplifier les notations nous considérons que l'ensemble P consiste en les t premiers utilisateurs de U, c'est-à-dire que $P = \{1, \cdots, t\} = [t]$.

Schéma d'identification de cercle à seuil. Un schéma d'identification de cercle à seuil consiste en un triplet d'algorithmes $(\mathsf{RS{-}init}, \mathsf{RS{-}gen}, (L \leftrightarrow \mathcal{V}))$:

- $\mathsf{param} \leftarrow \mathsf{RS{-}init}(1^n)$: $\mathsf{RS{-}init}$ est un algorithme probabiliste polynomial d'initialisation, il prend en entrée un paramètre de sécurité 1^n et retourne un paramètre public param.
- $(pk_i, sk_i) \leftarrow \mathsf{RS{-}gen}(\mathsf{param}, i)$: $\mathsf{RS{-}gen}$ est un algorithme probabiliste polynomial qui prend en entrée le paramètre param et un entier i et retourne une paire de clés (pk_i, sk_i).
- $(L \leftrightarrow \mathcal{V})$: est un protocole interactif entre un leader L élu parmi un ensemble P contenant t prouveurs et un vérificateur $\mathcal{V}$. Les deux protagonistes ont comme paramètres partagés le paramètre public param et l'ensemble des clés publiques de tout les utilisateurs. A la fin de l'interaction, le vérificateur retourne $\mathsf{desc} \in \{0, 1\}$, qui vaut 1 si ce dernier accepte et 0 s'il rejette.

3.2.2 Sécurité

Nous proposons un schéma d'identification de cercle à seuil qui est basé sur une preuve de connaissance sans divulgation. Il permet à t parmi N prouveur de s'identifier auprès d'un vérificateur d'une manière anonyme.

Preuve de connaissance sans divulgation : Nous montrons que la preuve de connaissance sous-jacente à notre protocole est une preuve de connaissance qu'un leader L connait un ensemble de t secrets différents. Nous prouvons que le protocole ne divulgue aucune information sur l'ensemble des secrets pendant son exécution.

Anonymat : L'anonymat permet de s'assurer que l'on ne peut pas connaître les identités des prouveurs à partir de leurs transcriptions quand ils exécutent le protocole.

Nous introduisons une définition analogue à la définition d'anonymat pour les schémas d'identification de cercle (voir Définition 3.3 pour plus de détails). Par conséquent, un schéma d'identification de cercle à seuil est anonyme contre la révélation totale des clés si tout attaquant ne peut pas distinguer entre deux ensembles distincts de t prouveurs parmi N en regardant leurs transcriptions. Nous donnons une formalisation de cette notion dans la définition suivante.

Définition 3.4. *Soit un schéma d'identification de cercle défini par les algorithmes* $(\mathsf{RS}-\mathsf{init}, \mathsf{RS}-\mathsf{gen}, (L \leftrightarrow \mathcal{V}))$. *On considère le jeu suivant entre un challengeur* $\mathcal{C}$ *et un attaquant* $\mathcal{A}$.

Phase de préparation : *Soit* n *le paramètre de sécurité. Le challengeur envoie à l'attaquant* $(1^n, \mathsf{param} \leftarrow \mathsf{RS}-\mathsf{init}(1^n))$.

Phase de challenge : *Dans cette phase* $\mathcal{A}$ *envoie à* $\mathcal{C}$ *un challenge* $(R = \{\boldsymbol{y}_i\}_{i\in[N]}$, $S_0 = (sk_{i_1}, \cdots, sk_{i_t})$, $S_1 = (sk_{j_1}, \cdots, sk_{j_t}))$ *comme challenge au challengeur* $\mathcal{C}$, *avec* S_0, S_1 *deux ensembles distincts des clés secrètes valides, tels que leurs clés publiques correspondantes sont dans l'ensemble* R. *Ensuite le challengeur choisit un bit aléatoire* $b \in \{0,1\}$ *et il exécute le protocole avec l'attaquant, qui joue le rôle d'un vérificateur, en utilisant les clés secrètes dans l'ensemble* S_b.

Finalement l'attaquant retourne un bit b' *et il gagne le jeu si* $b' = b$.

Un schéma d'identification de cercle à seuil satisfait la propriété d'anonymat contre la révélation totale des clés si la probabilité qu'un attaquant puisse gagner ce jeu est négligeable en le paramètre de sécurité n.

3.2.3 Schéma de Cayrel-Lindner-Rückert-Silva

Dans cette partie, nous décrivons un schéma d'identification de cercle à seuil proposé par Cayrel, Lindner, Rückert et Silva [CLRS10b], que nous nommons TCLRS. Ce schéma se base sur le schéma d'identification en cinq passes CLRS (voir Annexe A) proposé par les mêmes auteurs en 2010 [CLRS10a].

Description. Soient n un paramètre de sécurité et q, m deux entiers qui dépendent de n. Chaque utilisateur $i \in U$ possède une paire de clés $(pk_i, sk_i) = (\boldsymbol{A}_i, \boldsymbol{x}_i)$ avec $\boldsymbol{A}_i \in \mathbb{Z}_q^{n\times(m+1)}$, $\boldsymbol{x}_i \in \{0,1\}^{m+1}$, $w_H(\boldsymbol{x}_i) = m/2 + 1$ et $\boldsymbol{A}_i\boldsymbol{x}_i = 0 \bmod q$. Avant d'exécuter le protocole, les t prouveurs dans P se mettent d'accord sur un leader L parmi eux.

Soit $\boldsymbol{A}$ la matrice définie de la manière suivante :

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_1 & 0 & \cdots & 0 \\ 0 & \boldsymbol{A}_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \boldsymbol{A}_N \end{bmatrix}$$

En utilisant le protocole interactif suivant, le vérificateur $\mathcal{V}$ obtient une preuve que t utilisateurs de l'ensemble U ont participé à la génération d'un vecteur $\boldsymbol{x} \in \{0,1\}^{N(m+1)}$ tel que $\boldsymbol{A}\boldsymbol{x} = 0 \bmod q$ et $w_H(\boldsymbol{x}) = t(m/2+1)$. Nous rappelons brièvement deux types de permutations de blocs utilisées dans le protocole. Soient n, M deux entiers, on a :

— une permutation constante de n-blocs est une permutation Φ qui permute M blocs de taille n par bloc.
— une permutation de n-blocs Π est définie par la composition de deux permutation Φ et π que nous notons par $\Pi = \Phi \circ \pi$, avec $\pi = (\pi_1, \cdots, \pi_M)$ telle que $\pi_i \in S_n$ pour tout $i \in [n]$ et Φ est une permutation constante de n-blocs.

$(L \leftrightarrow \mathcal{V})$: Les paramètres en commun entre le leader et le vérificateur sont la matrice $\boldsymbol{A}$ et une constante égale à $m/2+1$ qui représente le poids de Hamming de chaque clé secrète.

1. [Étape d'engagement]
   — Chaque prouveur i dans l'ensemble P, choisi $\pi_i \leftarrow S_{m+1}$, $\boldsymbol{u}_i \leftarrow \mathbb{Z}_q^{m+1}$ et il envoie $c_{0,i}$ et $c_{1,i}$ à L tels que :

$$c_{0,i} \leftarrow \mathsf{COM}(\pi_i \| \boldsymbol{A}_i \boldsymbol{u}_i), c_{1,i} \leftarrow \mathsf{COM}(\pi_i(\boldsymbol{u}_i) \| \pi_i(\boldsymbol{x}_i)).$$

   — Pour $i \in U \setminus P$, le leader L fait la même chose mais avec $\boldsymbol{x}_i \leftarrow 0$.
   — L choisit une permutation constante aléatoire Φ de n-blocs sur N blocs.
   — L calcule les engagements $C_0 = \mathsf{COM}(\Phi \| c_{0,1} \| \cdots \| c_{0,N})$ et $C_1 = \mathsf{COM}(\Phi(c_{1,1}, \cdots, c_{1,N}))$ et les envoie à $\mathcal{V}$.
2. [Étape de challenge] $\mathcal{V}$ choisit $\alpha \leftarrow \mathbb{Z}_q$ et l'envoie à L. A son tour L envoie α à tous les prouveurs de l'ensemble P.
3. [Étape de réponse]
   — Chaque prouveur de l'ensemble P, calcule $\boldsymbol{g}_i \leftarrow \pi_i(\boldsymbol{u}_i + \alpha \boldsymbol{x}_i)$
   — Pour $i \in U \setminus P$, le leader fait la même chose mais avec $\boldsymbol{x}_i \leftarrow 0$.
   — L envoie $\boldsymbol{c} = \Phi(\boldsymbol{g}_1, \cdots, \boldsymbol{g}_N)$ à $\mathcal{V}$.
4. [Étape de challenge] $\mathcal{V}$ choisit $b \leftarrow \{0,1\}$ et l'envoie à L. A son tour L l'envoie à tous les prouveurs de l'ensemble P.
5. [Étape de réponse]
   — Si $b = 0$, alors

- Chaque prouveur i dans P envoie π_i à L.
- Soit $\pi = (\pi_1, \cdots, \pi_N)$.
- L envoie $\Pi = \Phi \circ \pi$ à $\mathcal{V}$.
- $\mathcal{V}$ vérifie que Π est bien une permutation par blocs et que $C_0 = \mathsf{COM}(\Phi \| \boldsymbol{A}\Pi^{-1}(\boldsymbol{c}))$.

— Si $b = 1$, alors
- Chaque prouveur i dans P envoie $\pi_i(\boldsymbol{x}_i)$ à L.
- Soit $\Pi(\boldsymbol{x}) = \Phi(\pi_1(\boldsymbol{x}_1), \cdots, \pi_N(\boldsymbol{x}_N))$.
- L envoie $\Pi(\boldsymbol{x})$ à $\mathcal{V}$.
- $\mathcal{V}$ vérifie si $\Pi(\boldsymbol{x}) \in \{0,1\}^{N\times(m+1)}$, $w_H(\boldsymbol{x}) = t(m/2+1)$ et que $C_1 = \mathsf{COM}(\boldsymbol{c} - \alpha\Pi(\boldsymbol{x}) \| \Pi(\boldsymbol{x}))$.

$\mathcal{V}$ retourne $\mathsf{desc} = 1$ si les tests sont valides, sinon il retourne $\mathsf{desc} = 0$.

Nous renvoyons le lecteur à [CLRS10b] pour le détail des preuves de sécurité.

3.2.4 Notre schéma d'identification de cercle à seuil

3.2.4.1 Aperçu

Soit U un cercle de N membres dont t utilisateurs veulent s'identifier d'une manière collective et anonyme auprès d'un vérificateur $\mathcal{V}$. Chaque utilisateur i utilise l'algorithme de génération des clés $\mathsf{RS}-\mathsf{gen}$ (voir la section 3.2.4.2) pour obtenir une paire de clés $(pk_i, sk_i) = (\boldsymbol{y}_i, \boldsymbol{x}_i)$. Une clé publique des utilisateurs consiste en la matrice $\boldsymbol{A}$, l'ensemble des clés publiques $\{\boldsymbol{y}_1, \cdots, \boldsymbol{y}_N\}$ et un entier fixe qui représente le poids de Hamming de la clé secrète des utilisateurs. Une information secrète est partagée entre les t prouveurs qui consiste en une permutation Σ de l'ensemble $[N]$ choisie au début par un leader L parmi l'ensemble des prouveurs.

L'idée principale de notre protocole est que chacun des t prouveurs calcule une instance du schéma d'identification de cercle (voir section 3.1.3.2) en utilisant comme entrée les matrices $\boldsymbol{A}$ et $\boldsymbol{M}$ (avec $\boldsymbol{M} = [\boldsymbol{y}_1 \| \cdots \| \boldsymbol{y}_N]$), sa propre clé secrète et la permutation Σ. Le leader L est chargé de collecter les résultats, y compris le sien, et d'exécuter par la suite un protocole interactif avec le vérificateur $\mathcal{V}$.

La propriété d'anonymat est héritée du schéma de base. En effet, notre protocole n'est qu'une combinaison de t instances du schéma d'identification de cercle. Afin de montrer au vérificateur que exactement t prouveurs ont participé dans l'interaction, nous avons rajouté un test supplémentaire pendant la phase de vérification. L'idée est très simple, le leader doit montrer que les vecteurs $\boldsymbol{e}_i$ (ce sont les vecteurs qui permettent l'identification d'un prouveur via sa clé publique $\boldsymbol{y}_i$) utilisés dans le protocole sont différents. Par conséquent, le vérificateur vérifie si pour tout $i \in [t]$, $\boldsymbol{w}_i = \Sigma(\boldsymbol{e}_i)$ est un vecteur dans $\{0,1\}^N$ et que $w_H(\sum_{i=1}^{t} \boldsymbol{w}_i) = t$.

La preuve de connaissance donnée par L consiste en un ensemble $S = \{(\boldsymbol{s}_i, \boldsymbol{z}_i)\}_{i\in[t]}$

qui satisfait les propriétés suivantes :
— avec pour $i \in [t]$, $(\boldsymbol{s}_i, \boldsymbol{z}_i) \in \{0,1\}^m \times \in \{0,1\}^N$ et $w_H(\boldsymbol{s}_i) = m/2$, $w_H(\boldsymbol{z}_i) = 1$
— $\boldsymbol{A}\boldsymbol{s}_i - \boldsymbol{M}\boldsymbol{z}_i = 0 \bmod q$.
— $w_H(\sum_i \boldsymbol{z}_i) = t$.
Par conséquent n'importe quel ensemble de t prouveurs honnêtes qui exécutent le protocole satisfait ces propriétés.

3.2.4.2 Description

Nous utilisons les mêmes définitions que le schéma TCLRS pour les ensembles U et P. Soient n le paramètre de sécurité et q, m deux entiers tels que $q = \mathsf{poly(n)}$ et $m \geq 10n \log q$.

$\boldsymbol{A} \leftarrow \mathsf{RS{-}init}(1^n)$: Étant donné un paramètre de sécurité n. Choisir une matrice $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$ d'une façon uniformément aléatoire.

$(\boldsymbol{y}_i, \boldsymbol{x}_i) \leftarrow \mathsf{RS{-}gen}(\boldsymbol{A}, i)$: Étant donnés une matrice $\boldsymbol{A}$ et un entier i correspondant à l'indice de l'utilisateur dans U, choisir une paire de clés en utilisant la procédure suivante :
— Soit $\boldsymbol{x}_i \leftarrow \{0,1\}^m$, tel que $w_H(\boldsymbol{x}_i) = m/2$.
— Soit $\boldsymbol{y}_i = \boldsymbol{A}\boldsymbol{x}_i \bmod q$.

$(L \leftrightarrow \mathcal{V})$: Les paramètres en commun entre le leader et le vérificateur sont les paramètres publics $(\boldsymbol{A}, \boldsymbol{M})$ avec $\boldsymbol{M} = [\boldsymbol{y}_1 \| \cdots \| \boldsymbol{y}_N]$.

1. [Étape d'engagement]
   — Le leader choisit une permutation $\Sigma \leftarrow S_N$ et il l'envoie à tous les prouveurs dans P.
   — Chaque prouveur i dans l'ensemble P, choisit $\pi_i \leftarrow S_m, \boldsymbol{u}_i \leftarrow \mathbb{Z}_q^m, \boldsymbol{u}'_i \leftarrow \mathbb{Z}_q^N$ et il envoie $c_{0,i}$ et $c_{1,i}$ à L tels que :

   $$c_{0,i} \leftarrow \mathsf{COM}(\pi_i \| \Sigma \| \boldsymbol{A}\boldsymbol{u}_i - \boldsymbol{M}\boldsymbol{u}'_i), c_{1,i} \leftarrow \mathsf{COM}(\pi_i(\boldsymbol{u}_i) \| \Sigma(\boldsymbol{u}'_i) \| \pi_i(\boldsymbol{x}_i) \| \Sigma(\boldsymbol{e}_i))$$

   — L calcule les engagements $C_0 = \mathsf{COM}(c_{0,1} \| \cdots \| c_{0,t})$ et $C_1 = \mathsf{COM}(c_{1,1} \| \cdots \| c_{1,t})$.
   — L envoie C_0 et C_1 à $\mathcal{V}$.
2. [Étape de challenge] $\mathcal{V}$ choisit $\alpha \leftarrow \mathbb{Z}_q$ et l'envoie à L.
3. [Étape de réponse]
   — L envoie α à tous les prouveurs dans P.
   — Chaque prouveur dans l'ensemble P, envoie $\boldsymbol{g}$ et $\boldsymbol{g}'$ tels que

   $$\boldsymbol{g}_i \leftarrow \pi_i(\boldsymbol{u}_i + \alpha \boldsymbol{x}_i), \boldsymbol{g}'_i \leftarrow \Sigma(\boldsymbol{u}'_i + \alpha \boldsymbol{e}_i)$$

   — Pour tout $i \in [t]$, L envoie $\boldsymbol{c}_i = \boldsymbol{g}_i$ et $\boldsymbol{d}_i = \boldsymbol{g}'_i$ à $\mathcal{V}$.
4. [Étape de challenge] $\mathcal{V}$ choisit $b \leftarrow \{0,1\}$ et l'envoie à L.
5. [Étape de réponse]

— Si $b = 0$, alors
  — Chaque prouveur i dans P envoie π_i à L.
  — L envoie, $\phi = \Sigma, \psi_1 = \pi_1, \cdots, \psi_t = \pi_t$ à $\mathcal{V}$.
  — Pour $i \in [t], \mathcal{V}$ calcule

$$c_i = \mathsf{COM}(\psi_i \| \phi \| \boldsymbol{A}\psi_i^{-1}(\boldsymbol{c}_i) - \boldsymbol{M}\phi^{-1}(\boldsymbol{d}_i)).$$

  — $\mathcal{V}$ vérifie si $C_0 = \mathsf{COM}(c_1 \| \cdots \| c_t)$.
— Si $b = 1$, alors
  — Chaque prouveur i dans P envoie $\Sigma(\boldsymbol{e}_i)$ et $\pi_i(\boldsymbol{x}_i)$ à L.
  — L envoie, $\boldsymbol{v}_1 = \pi_1(\boldsymbol{x}_1), \cdots, \boldsymbol{v}_t = \pi_t(\boldsymbol{x}_t)$ et $\boldsymbol{w}_1 = \Sigma(\boldsymbol{e}_1), \cdots, \boldsymbol{w}_t = \Sigma(\boldsymbol{e}_t)$ à $\mathcal{V}$.
  — Pour $i \in [t], \mathcal{V}$ calcule

$$c_i = \mathsf{COM}(\boldsymbol{c}_i - \alpha\boldsymbol{v}_i \| \boldsymbol{d}_i - \alpha\boldsymbol{w}_i \| \boldsymbol{v}_i \| \boldsymbol{w}_i).$$

  — $\mathcal{V}$ vérifie si
    — pour tout $i \in [t]$, $\boldsymbol{v}_i \in \{0,1\}^m$ et $w_H(\boldsymbol{v}_i) = m/2$,
    — pour tout $i \in [t]$, $\boldsymbol{w}_i \in \{0,1\}^N$, $w_H(\boldsymbol{w}_i) = 1$,
    — $w_H(\sum_{i=1}^{t} \boldsymbol{w}_i) = t$,
    — $C_1 = \mathsf{COM}(c_1 \| \cdots \| c_t)$.

$\mathcal{V}$ retourne $\mathsf{desc} = 1$ si les tests sont valides, sinon il retourne $\mathsf{desc} = 0$.

Complétude : Si P est un ensemble qui contient que des prouveurs honnêtes, alors quels que soient les challenges envoyés par le vérificateur, les prouveurs dans P peuvent toujours lui fournir des réponses qui sont valides. En effet, la connaissance des valeurs $(\boldsymbol{x}_i, \boldsymbol{e}_i)$ permet de calculer correctement les valeurs des engagements $c_{0,i}$, $c_{1,i}$ ainsi que les valeurs $\boldsymbol{g}_i, \boldsymbol{g}'_i$. Pendant la dernière étape : si $b = 0$, la divulgation de toutes les permutations permettra au vérificateur de retrouver l'engagement C_0 ; pour le cas $b = 1$, les prouveurs envoient leurs secrets masqués avec des permutations qui sont connues uniquement par eux. Ceci permet au vérificateur de retrouver l'engagement C_1 et de vérifier que t clés secrètes différentes ont été utilisées.

3.2.4.3 Analyse de sécurité

Théorème 3.3 (Sans divulgation)**.** *Si le schéma d'engagement satisfait la propriété d'indistinguabilité (Hiding), alors notre schéma est sans divulgation.*

Démonstration. Nous construisons un simulateur $\mathcal{S}$ de même style de celui du schéma d'identification de cercle (voir la preuve du Théorème 3.1). $\mathcal{S}$ prend comme entrées les matrices $\boldsymbol{A}$ et $\boldsymbol{M}$ et a accès à un vérificateur malhonnête $\tilde{\mathcal{V}}$ qui fournit des challenges $(\alpha \in \mathbb{Z}_q, b \in \{0,1\})$. Ce simulateur retourne une distribution statistiquement proche de la vue d'un vérificateur honnête $\mathcal{V}$ lorsque il interagit avec un leader honnête L.

Le simulateur doit deviner le challenge b, avant d'interagir avec $\tilde{\mathcal{V}}$, nous supposons que $\mathcal{S}$ devine correctement le challenge :

— Si $b = 0$, le simulateur choisit au hasard un ensemble T de cardinal t tel que $T := \{\boldsymbol{y}_{i_1}, \cdots, \boldsymbol{y}_{i_t}\} \subset \{\boldsymbol{y}_i\}_{i\in[N]}$, ensuite il calcule $\boldsymbol{s}_1, \cdots, \boldsymbol{s}_t$ tels que pour tout $j \in [t]$, $\boldsymbol{A}\boldsymbol{s}_j = \boldsymbol{y}_{i_j} \bmod q$. Le simulateur construit les vecteurs $\boldsymbol{e}_{i_1}, \boldsymbol{e}_{i_t}$ qui correspondent au vecteur de l'ensemble T. Puis il calcule C_0, C_1 comme dans le protocole en utilisant $\{(\boldsymbol{s}_j, \boldsymbol{e}_{i_j})\}_{j\in[t]}$. Après avoir reçu α de la part du vérificateur $\tilde{\mathcal{V}}$, le simulateur calcule $\{\boldsymbol{g}_i; \boldsymbol{g}'_i\}_{i\in[t]}$. Grâce à la propriété d'indistinguabilité du schéma d'engagement, nous avons que la distribution de (C_0, C_1) est indistinguable de la distribution d'un couple qui aurait pu être générée par un leader honnête L.

— Si $b = 1$, le simulateur choisit t vecteurs $\boldsymbol{s}_1, \cdots, \boldsymbol{s}_t \in \{0,1\}^m$ différents tels que $w_H(\boldsymbol{s}_i) = m/2$ pour $i \in [t]$ et t vecteurs $\boldsymbol{e}_1, \cdots, \boldsymbol{e}_t \in \{0,1\}^\ell$ différents tels que $w_H(\boldsymbol{e}_i) = 1$ pour $i \in [t]$. Ensuite il calcule les deux challenges C_0 et C_1, après la réception de α de la part de $\tilde{\mathcal{V}}$, il calcule $\{\boldsymbol{g}_i; \boldsymbol{g}'_i\}_{i\in[t]}$ comme dans le protocole. Nous obtenons une vérification valide pour $b = 1$.

La probabilité que le simulateur produise une simulation correcte est de $1/2$. Lorsque le simulateur devine correctement le challenge b du vérificateur, il produit une transcription statistiquement proche d'une interaction réelle entre un leader L et un vérificateur $\mathcal{V}$. Le simulateur enregistre les valeurs de C_0, C_1, α, $\boldsymbol{g}, \boldsymbol{g}'$ et la réponse au challenge b lorsqu'il devine correctement la valeur de b. Sinon il relance le vérificateur $\tilde{\mathcal{V}}$. □

Théorème 3.4 (Preuve de connaissance)**.** *Si le schéma d'engagement* COM *satisfait la propriété d'engagement (binding), alors notre protocole est une preuve de connaissance. En particulier, s'il existe un prouveur malhonnête $\tilde{L}$ (algorithme probabiliste polynomial) qui convainc le vérificateur du protocole avec probabilité $1/2 + \varepsilon$, avec $\varepsilon = \varepsilon(n) > 0$ non négligeable, alors il existe un extracteur $\mathcal{E}$ (algorithme probabiliste polynomial) qui retourne un ensemble $S = \{\boldsymbol{s}_1, \cdots, \boldsymbol{s}_t\}$ tel qu'il existe t clés publiques $\{\boldsymbol{y}_{j_1}, \cdots, \boldsymbol{y}_{j_t}\} \subset \{\boldsymbol{y}_i\}_{i\in[N]}$ avec pour $i \in [t]$:*

— $\boldsymbol{s}_i \in \{0,1\}^m$ *et* $w_H(\boldsymbol{s}_i) = m/2$

— $\boldsymbol{A}\boldsymbol{s}_i = \boldsymbol{y}_{j_i} \bmod q$.

Démonstration. Soit $\tilde{L}$ un prouveur malhonnête (qui représente un leader malhonnête) dans le sens où il ne connait aucune clé secrète valide. Nous montrons comment construire un extracteur $\mathcal{E}$ qui peut calculer t témoins différents en utilisant $\tilde{L}$. L'extracteur joue le rôle d'un vérificateur comme dans le protocole et interagit avec le prouveur malhonnête $\tilde{L}$. Au début $\mathcal{E}$ reçoit (C_0, C_1) comme engagement de la part de $\tilde{L}$. Ensuite $\mathcal{E}$ envoie à $\tilde{L}$, les $2q$ challenges possibles sur le même engagement (C_0, C_1) et il enregistre la transcription obtenue à chaque fois.

Par construction notre protocole est la combinaison de t instances du schéma d'identification de cercle présenté dans la section 3.1.3. Donc l'attaquant $\tilde{L}$ peut utiliser les stratégies de triche décrites dans la première partie de la preuve du

Théorème 3.2, pour réussir à casser le protocole d'identification du cercle à seuil. Plus précisément, $\tilde{L}$ construit t instances différentes. D'après la même preuve nous savons que $\tilde{L}$ peut répondre correctement à $q+1$ requêtes parmi $2q$, ceci implique que la probabilité de triche pour un seul tour du protocole est $(q+1)/2q \approx 1/2$ pour $q = \mathsf{poly}(n)$.

Supposons que $\tilde{L}$ parvienne à répondre correctement à $q+2$ requêtes parmi les $2q$ requêtes possibles envoyées par $\mathcal{E}$, pour un engagement (C_0, C_1). En utilisant le principe des tiroirs, nous déduisons que, l'attaquant $\tilde{L}$ peut répondre correctement pour deux challenges différents (α_1, α_2) dans la première phase de challenge et pour n'importe quelle valeur de b dans la deuxième phase de challenge.

Par conséquent, $\mathcal{E}$ obtient les transcriptions suivantes :

$$\mathsf{T}_1 = \{(C_0, C_1), \alpha_1, (\boldsymbol{c}_{1,i}, \boldsymbol{d}_{1,i}; i \in [t]), 0, (\phi_1, \psi_{1,i}; i \in [t])\} \tag{3.6}$$

$$\mathsf{T}_2 = \{(C_0, C_1), \alpha_1, (\boldsymbol{c}_{1,i}, \boldsymbol{d}_{1,i}; i \in [t]), 1, (\boldsymbol{v}_{1,i}, \boldsymbol{w}_{1,i}; i \in [t])\} \tag{3.7}$$

$$\mathsf{T}_3 = \{(C_0, C_1), \alpha_2, (\boldsymbol{c}_{2,i}, \boldsymbol{d}_{2,i}; i \in [t]), 0, (\phi_2, \psi_{2,i}; i \in [t])\} \tag{3.8}$$

$$\mathsf{T}_4 = \{(C_0, C_1), \alpha_2, (\boldsymbol{c}_{2,i}, \boldsymbol{d}_{2,i}; i \in [t]), 1, (\boldsymbol{v}_{2,i}, \boldsymbol{w}_{2,i}; i \in [t])\} \tag{3.9}$$

Puisque chaque transcription représente une exécution valide, $\mathcal{E}$ obtient ou bien deux collisions pour la fonction d'engagement COM ou bien $\mathcal{E}$ obtient les équations suivantes :

a. $\phi_1 = \phi_2, \forall i \in [t]; \psi_{1,i} = \psi_{2,i}$,

b. $\forall i \in [t]; \boldsymbol{A}\psi_{1,i}^{-1}(\boldsymbol{c}_{1,i}) - \boldsymbol{M}\phi_1^{-1}(\boldsymbol{d}_{1,i}) = \boldsymbol{A}\psi_{2,i}^{-1}(\boldsymbol{c}_{2,i}) - \boldsymbol{M}\phi_2^{-1}(\boldsymbol{d}_{2,i})$

c. $\forall i \in [t]; \boldsymbol{c}_{1,i} - \alpha_1\boldsymbol{v}_{1,i} = \boldsymbol{c}_{2,i} - \alpha_2\boldsymbol{v}_{2,i}$ et $\boldsymbol{d}_{1,i} - \alpha_1\boldsymbol{w}_{1,i} = \boldsymbol{d}_{2,i} - \alpha_2\boldsymbol{w}_{2,i}$.

d. $\forall i \in [t]; \boldsymbol{v}_{1,i} = \boldsymbol{v}_{2,i}$, $\boldsymbol{w}_{1,i} = \boldsymbol{w}_{2,i}$

e. $\forall i \in [t]; w_H(\boldsymbol{v}_{1,i}) = w_H(\boldsymbol{v}_{2,i}) = m/2$ avec $\boldsymbol{v}_{1,i}, \boldsymbol{v}_{2,i} \in \{0,1\}^m$.

f. $\forall i \in [t]; \boldsymbol{w}_{1,i}, \boldsymbol{w}_{2,i} \in \{0,1\}^N$ et $\sum_{i=1}^t w_H(\boldsymbol{w}_{1,i}) = \sum_{i=1}^t w_H(\boldsymbol{w}_{2,i}) = t$.

Où les équations dans (a, b) sont obtenues grâce aux équations (3.6) et (3.8). Les équations dans $(\mathsf{c}, \mathsf{d}, \mathsf{e}, \mathsf{f})$ sont obtenues grâce aux équations (3.7) et (3.9).

A partir des équations dans (c) nous avons que $\boldsymbol{c}_{2,i} = \boldsymbol{c}_{1,i} - \alpha_1\boldsymbol{v}_{1,i} + \alpha_2\boldsymbol{v}_{2,i}$ et $\boldsymbol{d}_{2,i} = \boldsymbol{d}_{1,i} - \alpha_1\boldsymbol{w}_{1,i} + \alpha_2\boldsymbol{w}_{2,i}$ pour tout $i \in [t]$. En remplaçant $\boldsymbol{c}_{2,i}$ et $\boldsymbol{d}_{2,i}$ dans l'équation (b), $\mathcal{E}$ obtient que pour tout $i \in [t]$:

$$\boldsymbol{A}\psi_{1,i}^{-1}(\boldsymbol{c}_{1,i}) - \boldsymbol{M}\phi_1^{-1}(\boldsymbol{d}_{1,i}) = \boldsymbol{A}\psi_{1,i}^{-1}(\boldsymbol{c}_{1,i} - \alpha_1\boldsymbol{v}_{1,i} + \alpha_2\boldsymbol{v}_{2,i}) - \boldsymbol{M}\phi_1^{-1}(\boldsymbol{d}_{1,i} - \alpha_1\boldsymbol{w}_{1,i} + \alpha_2\boldsymbol{w}_{2,i}).$$

En utilisant les égalités dans (d), ceci implique que pour tout $i \in [t]$:

$$(\alpha_2 - \alpha_1)(\boldsymbol{A}\psi_{1,i}^{-1}(\boldsymbol{v}_{1,i}) - \boldsymbol{M}\phi_1^{-1}(\boldsymbol{w}_{1,i})) = 0.$$

Puisque $\alpha_1 \neq \alpha_2$, nous avons que $\boldsymbol{A}\psi_{1,i}^{-1}(\boldsymbol{v}_{1,i}) - \boldsymbol{M}\phi_1^{-1}(\boldsymbol{w}_{1,i}) = 0 \bmod q$ pour tout $i \in [t]$. Pour tout $i \in [t]$, notons par $\boldsymbol{s}_i$ les vecteurs $\psi_{1,i}^{-1}(\boldsymbol{v}_{1,i})$ d'après les équations

(e,f), il existe $\{\boldsymbol{y}_{j_1},\cdots,\boldsymbol{y}_{j_t}\}\subset\{\boldsymbol{y}_i\}_{i\in[N]}$ tel que pour $i\in[t]$, $\boldsymbol{A}\boldsymbol{s}_i-\boldsymbol{y}_{j_i}=0 \bmod q$, $\boldsymbol{s}_i\in\{0,1\}^m$ et $w_H(\boldsymbol{s}_i)=m/2$.

Donc si le prouveur $\tilde{L}$ arrive à convaincre $\mathcal{E}$ avec une probabilité supérieur ou égale à $\frac{q+1}{2q}+\varepsilon(n)$ (avec, $\varepsilon(n)$ non négligeable) dans chaque tour du protocole, alors $\mathcal{E}$ peut extraire un ensemble $S=\{\boldsymbol{s}_1,\cdots,\boldsymbol{s}_t\}$ qui satisfait les conditions du théorème. □

Remarque 3.1. *Le protocole doit être exécuté $t=\omega(\log n)$ fois pour obtenir une faible probabilité de fraude.*

Théorème 3.5 (Anonymat)**.** *Si le schéma d'engagement satisfait la propriété d'indistinguabilité, alors notre schéma vérifie l'anonymat contre la révélation totale des clés.*

Démonstration. En combinant le Théorème 3.3 et la Proposition 1.4, nous obtenons que notre schéma est à témoin indistinguable.

Supposons que l'attaquant $\mathcal{A}$ envoie $(R=\{\boldsymbol{y}_i\}_{i\in[N]},\ S_0=(sk_{i_1},\cdots,sk_{i_t}),\ S_1=(sk_{j_1},\cdots,sk_{j_t}))$ comme challenge au challengeur $\mathcal{C}$, avec S_0,S_1 deux ensembles distincts tels que leurs clés publiques correspondantes sont dans l'ensemble R. Nous montrons que les transcriptions dans les deux cas, c'est-à-dire lorsque $\mathcal{C}$ exécute le protocole avec S_0 ou bien S_1, sont statistiquement indistinguables.

En effet, puisque notre schéma est à témoin indistinguable alors nous avons :

$$\mathsf{Vue}_{\mathcal{A}}\left[(\mathcal{C}(S_0)\leftrightarrow\mathcal{A}(S_0\|S_1))(R)\right]\approx_S \mathsf{Vue}_{\mathcal{A}}\left[(\mathcal{C}(S_1)\leftrightarrow\mathcal{A}(S_0\|S_1))(R)\right].$$

□

3.2.4.4 Coût et complexité

Comparaison avec le schéma TCLRS**.** Dans le schéma TCLRS, une transcription d'un tour du protocole consiste en :

$$(C_0,C_1;\alpha;\boldsymbol{c}=\Phi(\boldsymbol{g}_1,\cdots,\boldsymbol{g}_N);b;\mathsf{rps}_b),$$

avec C_0,C_1 deux vecteurs dans $\mathbb{Z}_q^n$, $\alpha\in\mathbb{Z}_q$, $\boldsymbol{g}_i\in\mathbb{Z}_q^m$ pour $i\in[N]$, $\Phi\in S_N$, $b\in\{0,1\}$ et rps_b un ensemble de permutations si $b=0$ et N vecteurs dans $\{0,1\}^m$ si $b=1$. Par conséquent, la taille de la transcription est en $O(N)$ avec $(\boldsymbol{g}_1,\cdots,\boldsymbol{g}_N)$ est la partie la plus coûteuse de taille $N\times m\log q$ bits. D'autre part, pour notre schéma présenté dans la section 3.2.4.2, une transcription a la forme suivante :

$$(C_0,C_1;\alpha;\boldsymbol{c}=(\boldsymbol{g}_1,\cdots,\boldsymbol{g}_t),\boldsymbol{d}=(\boldsymbol{g}'_1,\cdots,\boldsymbol{g}'_t);b;\mathsf{rps}_b),$$

avec C_0,C_1 deux vecteurs dans $\mathbb{Z}_q^n$, $\alpha\in\mathbb{Z}_q$, $\boldsymbol{g}_i\in\mathbb{Z}_q^m,\boldsymbol{g}'_i\in\mathbb{Z}_q^N$ pour $i\in[t]$, $b\in\{0,1\}$ et rps_b un ensemble de permutations si $b=0$ et t vecteurs dans $\{0,1\}^m$ de poids

de Hamming égal à $m/2$, t vecteurs dans $\{0,1\}^N$ de poids de Hamming égal à un dans le cas où $b = 1$. La partie coûteuse pour notre schéma est le couple $(\boldsymbol{c}, \boldsymbol{d})$, avec $\boldsymbol{c}$ de taille $t \times m \log q$ bits et $\boldsymbol{d}$ de taille $t \times N \log q$ bits. Par conséquent nous obtenons des tailles de communication plus faible que TCLRS si $t \times N < m$ tel que $m = 10n \log q$. En effet nous pouvons choisir un cercle de 100 utilisateurs et $t \in [100]$, avec $n = 2^9$ et $q = 2^{16}$ (nous nous référons aux travaux de Rücket et Schneider [RS10] pour ces valeurs de n et q).

Conclusion

Dans ce manuscrit, nous avons exposé nos travaux de recherche suivant deux axes. Le premier axe s'intéresse aux schémas de signature de cercle. Nous avons ainsi présenté un nouveau schéma de signature de cercle basé sur les réseaux. Ce schéma est le premier à être basé sur l'approche de la transformation de Fiat-Shamir. Ceci est intéressant d'un point de vue théorique pour deux raisons. La première est que c'est l'une des principales approches pour la construction des schémas de signature classique et que jusqu'à présent il n'y a pas de signature de cercle basée sur cette transformation. La deuxième raison est que notre schéma est une transformation du schéma de Lyubashevsky, ce qui implique que nous n'utilisons pas les fonctions à pré-image échantiollonable ou bien les preuves de connaissances sans divulgation. Du point de vue de la sécurité, notre démarche a été la suivante :

— En ce qui concerne l'anonymat, nous avons introduit une nouvelle formalisation de la notion d'anonymat pour les schémas de signature de cercle. Dans le jeu modélisant cette notion, nous donnons l'attaquant la possibilité de générer lui-même les clés secrètes et les paramètres publics du schéma. À la lumière des résultats récents sur le pourcentage élevé des paires de clés RSA non sûres [LHA+12], nous trouvons que ce résultat est une garantie rassurante pour les utilisateurs d'un schéma de signature de cercle. Nous avons fourni une preuve d'anonymat pour notre schéma suivant cette nouvelle définition d'anonymat. Dans un premier temps, nous avons montré que si la distance statistique entre deux distributions est négligeable en le paramètre de sécurité, alors le fait qu'un éventuel attaquant ai donné plusieurs éléments par distribution n'augmentera pas ses chances pour distinguer entre les deux distributions. Dans un deuxième temps, nous avons prouvé que si la distance statistique entre les distributions de deux signatures est négligeable en le paramètre de sécurité, alors le fait qu'un attaquant ait l'accès aux paramètres publics du schéma ainsi qu'aux clés publiques des utilisateurs n'augmentera pas la distance statistique. Finalement, nous avons démontré que la distance statistique entre les deux variables aléatoires modélisant deux signatures générées en utilisant notre schéma de signature est négligeable en le paramètre de sécurité.

— D'autre part, nous avons donné deux preuves d'inforgeabilité pour notre schéma. La première preuve consiste en l'inforgeabilité contre les attaques à sous-cercles choisis. En premier lieu, nous avons donné une preuve d'inforgeabilité du schéma de signature LSig09 lorsqu'il est instancié avec une fonction de hachage ayant $n^c \cdot m_u$ colonnes. Ensuite, nous avons montré que s'il existe un attaquant qui peut casser l'inforgeabilité de notre schéma contre les attaques à sous-cercles choisis en prenant comme challenge une fonction de hachage ayant $n^c \cdot m_u$ colonnes, alors il sera capable de casser le schéma de signature LSig09 défini avec la même fonction. La deuxième preuve consiste en l'inforgeabilité contre les attaques de corruption interne. Tout d'abord, nous avons proposé une version modifiée de notre schéma afin d'avoir l'inforgeabilité suivant cette notion. Ensuite nous avons donné une preuve d'anonymat et une analyse de la preuve d'inforgeabilité en nous inspirant de la preuve dans le cas d'inforgeabilité contre les attaques à sous-cercles choisis.

Un second axe de recherche a été de concevoir des schémas d'identification anonymes. Dans un premier temps, nous avons conçu un nouveau schéma d'identification de cercle et prouvé sa sécurité dans le modèle de sécurité proposé par Kawachi, Tanaka et Xagawa. Pour l'anonymat, nous avons tout d'abord montré que notre schéma est à témoin indistinguable. Ensuite, nous avons utilisé la définition de cette propriété pour montrer que notre schéma vérifie l'anonymat contre la révélation totale des clés. En ce qui concerne l'usurpation d'identité, nous avons prouvé que notre schéma est sûr contre l'usurpation d'identité avec des attaques parallèles à cercle choisi. En comparant les performances de notre schéma à celles de Kawachi et al., nous avons montré que notre schéma est plus efficace en terme de communication.

Dans un second temps, nous avons montré que la généralisation de notre schéma d'identification de cercle permet d'obtenir un schéma d'identification de cercle à seuil. Notre construction est une amélioration du schéma TCLRS proposé par Cayrel et al. en 2010. Du point de vue de la sécurité, nous avons montré que notre schéma permet de donner une preuve qu'un leader L connait un ensemble t secrets différents parmi N et qu'aucune information supplémentaire sur les secrets des utilisateurs n'est divulgué. Notre schéma est anonyme dans le sens où il est impossible pour un attaquant de connaître les identités des prouveurs à partir de leurs transcriptions.

Annexe A

Le schéma d'identification de Cayrel-Lindner-Rückert-Silva

Annexe A

Le schéma d'identification de Cayrel-Lindner-Rückert-Silva

Description. Soient n le paramètre de sécurité, q et m deux entiers tels que $q = \mathsf{poly}(n)$ et $m \geq 10n \log q$. La clé secrète du prouveur est un vecteur $\boldsymbol{x} \in \{0,1\}^m$ de poids de Hamming égal à $m/2$. La clé publique est un vecteur $\boldsymbol{y}$ lié à la clé secrète $\boldsymbol{x}$ par l'équation $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} \bmod q$, où $\boldsymbol{A}$ est une matrice connue par les deux parties.

Pendant chaque tour du protocole, le vérificateur examine une propriété parmi deux que le secret doit satisfaire. Plus précisément, une fois le vérificateur demande la vérification du poids de Hamming de $\boldsymbol{x}$, il vérifie s'il est bien de poids égal à $m/2$. Une autre fois, il demande la vérification de l'équation liant le secret $\boldsymbol{x}$ à la clé publique $\boldsymbol{y}$ par rapport à la matrice $\boldsymbol{A}$, c'est-à-dire il examine si le secret satisfait l'équation $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} \bmod q$.

$\mathsf{Id-init}(1^n)$: Le même que $\mathsf{Id-init}$ dans la section 1.6.3.

$\mathsf{Id-keygen}(\boldsymbol{A})$: Le même que $\mathsf{Id-keygen}$ dans la section 1.6.3.

$(\mathcal{P} \leftrightarrow \mathcal{V})$: Les paramètres en commun entre le prouveur et vérificateur sont $(\boldsymbol{A}, \boldsymbol{y})$. Le prouveur possède la clé secrète $\boldsymbol{x}$ comme entrée auxiliaire.

1. $\mathcal{P}$ choisit $\pi \leftarrow S_m$, $\boldsymbol{u} \leftarrow \mathbb{Z}_q^m$. $\mathcal{P}$ envoie c_0 et c_1 à $\mathcal{V}$ tels que
   — $c_0 \leftarrow \mathsf{COM}(\pi \| \boldsymbol{A}\boldsymbol{u})$
   — $c_1 \leftarrow \mathsf{COM}(\pi(\boldsymbol{u}) \| \pi(\boldsymbol{x}))$
2. $\mathcal{V}$ choisit $\alpha \leftarrow \mathbb{Z}_q$ et il l'envoie à $\mathcal{P}$.
3. $\mathcal{P}$ calcule $\boldsymbol{g} = \pi(\boldsymbol{u} + \alpha\boldsymbol{x})$ et il l'envoie à $\mathcal{V}$.
4. $\mathcal{V}$ envoie $b \leftarrow \{0,1\}$, à $\mathcal{P}$.
5. Si $b = 0$ alors
   $\mathcal{P}$ envoie π à $\mathcal{V}$ qui vérifie si $c_0 = \mathsf{COM}(\pi \| \boldsymbol{A}\pi^{-1}(\boldsymbol{g}) - \alpha\boldsymbol{y})$.
   Si $b = 1$ alors
   $\mathcal{P}$ envoie $\pi(\boldsymbol{x})$ à $\mathcal{V}$ qui vérifie si $w_H(\pi(\boldsymbol{x})) = m/2$ et $c_1 = \mathsf{COM}(\boldsymbol{g} -$

$\alpha\pi(\boldsymbol{x})\|\pi(\boldsymbol{x}))$.
$\mathcal{V}$ retourne desc $= 1$ si les tests sont valides, sinon il retourne desc $= 0$.

Sécurité. Pour montrer la sécurité contre les attaques parallèles, Cayrel et al. ont utilisé la technique que Kawachi et al. [KTX08] expliquée dans la section 1.6.3. Ceci signifie qu'un attaquant qui peut gagner le jeu suivant le modèle de sécurité contre les attaques parallèles, serait capable de résoudre une instance aléatoire du problème $\mathsf{SIS}_{n,q,m,\sqrt{m}}$. Nous renvoyons le lecteur à [CLRS10a] pour le détail des preuves de sécurité.

Annexe B

Preuve du Théorème 1.6

Sommaire

Annexe B

Preuve du Théorème 1.6

B.1 Le lemme de bifurcation général

Lemme B.1 (Lemme 1 dans [BN06]). *Soit $q \geq 1$ un entier et H un ensemble de taille $h \geq 2$. Soit A un algorithme randomisé qui prend en entrée $x, h_1, \cdots, h_q$ et qui retourne une paire dont le premier élément est un entier dans ensemble $\{0, \cdots, q\}$ et le second élément est une sortie latérale. Soit IG un algorithme randomisé qu'on appelle le générateur d'entrée (input generator). La probabilité d'acceptation de A, notée acc, est définie par la probabilité que $J \geq 1$ dans l'expérience suivante :*

$$x \leftarrow \mathsf{IG}; h_1, \cdots, h_q; (J, \sigma) \leftarrow \mathsf{A}(x, h_1, \cdots, h_q).$$

L'algorithme de bifurcation $\mathsf{F_A}$ associé à A est un algorithme randomisé qui prend x comme entrée et procède comme suit :

Algorithme $\mathsf{F_A}(x)$

1. *Choisir aléatoirement une valeur ρ pour A*
2. *Soient $h_1, \cdots, h_q \leftarrow H$*
3. *Soit $(I, \sigma) \leftarrow \mathsf{A}(x, h_1, \cdots, h_q; \rho)$*
4. *Si $I = 0$, alors retourner $(0, \varepsilon, \varepsilon)$*
5. *Soient $h'_I, \cdots, h'_q \leftarrow H$*
6. *Soit $(I', \sigma') \leftarrow \mathsf{A}(x, h_1, \cdots, h_{I-1}, h'_I, \cdots, h'_q; \rho)$*
7. *Si ($I = I'$ et $h_I \neq h'_I$) alors retourner $(1, \sigma, \sigma')$*
8. *Sinon retourner $(0, \varepsilon, \varepsilon')$*

Soit

$$\mathsf{frk} = \mathsf{Pr}\left[b = 1 : x \leftarrow \mathsf{IG}; (b, \sigma, \sigma') \leftarrow \mathsf{F_A}(x)\right].$$

Alors

$$\mathsf{frk} \geq \mathsf{acc} \cdot \left(\frac{\mathsf{acc}}{q} - \frac{1}{h}\right).$$

B.2 La preuve

On va montrer comment construire l'algorithme $\mathcal{B}$ qui utilise l'attaquant $\mathcal{F}$ pour résoudre le problème Col$(h, D_\times)$, pour une fonction de hachage choisie d'une manière aléatoire dans l'ensemble $\mathcal{H}(\mathcal{D}, D_\times, m)$.

Préparation : Étant donnée une fonction de hachage $h \in \mathcal{H}(\mathcal{D}, D_\times, m)$, on choisit une clé secrète $\hat{\mathbf{s}} \in D_{s,c}^m$ et on calcule $h(\hat{\mathbf{s}}) = \mathbf{S}$. Soient $\mathbf{r}_1, \cdots, \mathbf{r}_{q_H} \leftarrow D_{s,c}$ avec q_H le nombre de fois que l'oracle H_L peut être appelé par $\mathcal{F}$ durant le jeu. On choisit deux valeurs aléatoires : b_s pour l'oracle de signature et b_f pour l'attaquant $\mathcal{F}$. Ensuite, on initialise $\mathcal{B}$ en lui donnant $(h, \hat{\mathbf{s}}, b_s, b_f, \mathbf{r}_1, \cdots, \mathbf{r}_{q_H})$. À son tour, $\mathcal{B}$ initialise l'attaquant $\mathcal{F}$ en lui donnant la clé publique $(h, \mathbf{S})$ et b_f.

Interaction avec les oracles : $\mathcal{B}$ répond aux requêtes pour l'oracle aléatoire H_L et aux requêtes pour l'algorithme de signature de la manière suivante.

— *Requête pour l'oracle* H_L : la réponse de $\mathcal{B}$ est le premier élément $\mathbf{r}_i$ dans la liste $(\mathbf{r}_1, \cdots, \mathbf{r}_{q_H})$. L'algorithme $\mathcal{B}$, garde une liste avec toutes les requêtes à l'oracle H_L, par conséquent lorsque l'attaquant envoie une deuxième fois la même requête, $\mathcal{B}$ répond avec la même valeur envoyée précédemment.

— *Requête de signature* : $\mathcal{B}$ utilise la clé secrète $\hat{\mathbf{s}}$ et la valeur aléatoire b_s pour générer une signature valide. Pendant la génération de la signature, l'oracle H_L va être sollicité, $\mathcal{B}$ utilise les mêmes étapes de la phase de requête pour l'oracle H_L ci-dessus pour donner une réponse $\mathbf{r}_i$ de la liste $(\mathbf{r}_1, \cdots, \mathbf{r}_{q_H})$.

Génération de la signature : À un moment donné, $\mathcal{F}$ retourne un message μ et sa signature $(\hat{\mathbf{z}}, \mathbf{e})$ avec une probabilité δ, tels que $\mathbf{e} = H_L(h(\hat{\mathbf{z}}) - \mathbf{Se}, \mu)$. $\mathcal{B}$ retourne la même signature que $\mathcal{F}$.

Analyse : On commence par remarquer que si $\mathcal{F}$ n'a pas envoyé la requête $(h(\hat{\mathbf{z}}) - \mathbf{Se}, \mu)$ à H_L, alors la probabilité que $\mathcal{F}$ trouve $\mathbf{e}$ tel que $\mathbf{e} = H_L(h(\hat{\mathbf{z}}) - \mathbf{Se}, \mu)$ est $1/|D_{s,c}|$. Par conséquent, $\mathbf{e}$ doit être l'un des $\mathbf{r}_i$ avec une probabilité $1 - 1/|D_{s,c}|$, ce qui implique que la probabilité de succès de $\mathcal{F}$ à casser l'inforgeabilité telle que $\mathbf{e}$ soit l'un des $\mathbf{r}_i$ est au moins égale à $\delta - 1/|D_{s,c}|$. Soit $j \in [q_H]$ tel que $\mathbf{e} = r_j$. Par conséquent, on a deux possibilités : ou bien $\mathbf{r}_j$ a été généré durant une requête de signature, ou il a été généré par une requête directe à l'oracle aléatoire. On laisse le deuxième cas pour la fin de la preuve.

Supposons que l'algorithme de signature a envoyé une requête de la forme $(h(\hat{\mathbf{y}}'), \mu')$ à H_L. Soient $\mathbf{e} = \mathbf{r}_j$ la réponse de H_L et $\hat{\mathbf{z}}' = \hat{\mathbf{s}}\mathbf{e} + \hat{\mathbf{y}}'$ la signature du message μ'. Donc on a deux cas : soit $\hat{\mathbf{z}}' \in D_z^m$ ou $\hat{\mathbf{z}}' \notin D_z^m$. Si $\hat{\mathbf{z}}' \in D_z^m$, alors l'algorithme a retourné $\hat{\mathbf{z}}', \mathbf{e}$. Dans ce cas, si $\mathcal{F}$ retourne une signature forgée $(\mu, \hat{\mathbf{z}}, \mathbf{e})$, alors soit $\mu \neq \mu'$ ou $\hat{\mathbf{z}} \neq \hat{\mathbf{z}}'$ (ou bien $\mu \neq \mu'$ et $\hat{\mathbf{z}} \neq \hat{\mathbf{z}}'$), parce que sinon l'attaquant $\mathcal{F}$ est en train de retourner des paires message/signature qu'il a déjà

vu. Si $\mu \neq \mu'$, alors on a une collision sur l'oracle H_L, parce que

$$H_L(h(\hat{\mathbf{z}}) - \mathbf{Se}, \mu) = \mathbf{e} = H_L(h(\hat{\mathbf{z}}') - \mathbf{Se}, \mu') \tag{B.1}$$

Si $\mu = \mu'$, alors forcément $\hat{\mathbf{z}} \neq \hat{\mathbf{z}}'$ et

$$H_L(h(\hat{\mathbf{z}}) - \mathbf{Se}, \mu) = \mathbf{e} = H_L(h(\hat{\mathbf{z}}') - \mathbf{Se}, \mu), \tag{B.2}$$

on a deux cas : $h(\hat{\mathbf{z}}) - \mathbf{Se} \neq h(\hat{\mathbf{z}}') - \mathbf{Se}$ ou $h(\hat{\mathbf{z}}) - \mathbf{Se} = h(\hat{\mathbf{z}}') - \mathbf{Se}$. Dans le premier cas, on obtient une collision sur H_L et dans le deuxième cas on a $h(\hat{\mathbf{z}}) = h(\hat{\mathbf{z}}')$ ce qui implique une collision sur h. Puisque $\hat{\mathbf{z}}, \hat{\mathbf{z}}' \in D_z^m \subset D_\times^m$, on obtient une solution pour le problème $\mathsf{Col}(h, D_\times)$. Passons maintenant au cas où $\hat{\mathbf{z}}' \notin D_z^m$. Dans ce cas, l'attaquant $\mathcal{F}$ retourne une signature forgée valide $(\mu, \hat{\mathbf{z}}, \mathbf{e})$, tels que $\mu \neq \mu'$ ou $\hat{\mathbf{z}} \neq \hat{\mathbf{z}}'$ (ou bien $\mu \neq \mu'$ et $\hat{\mathbf{z}} \neq \hat{\mathbf{z}}'$), parce que $(\hat{\mathbf{z}}', \mathbf{e})$ n'est pas une signature du message μ puisque $\hat{\mathbf{z}}' \notin D_z^m$. Dans le cas où $\mu \neq \mu'$, d'après l'équation B.1, on obtient une collision sur H_L. De même, si $\mu = \mu'$ et $\hat{\mathbf{z}} \neq \hat{\mathbf{z}}'$, d'après l'équation B.2, on a une collisions sur H_L ou bien une solution pour le problème $\mathsf{Col}(h, D_\times)$. En effet, on a $h(\hat{\mathbf{z}}) = h(\hat{\mathbf{z}}')$ de plus on sait que $\hat{\mathbf{z}} \in D_z^m \subset D_\times^m$ et on a que $\hat{\mathbf{z}}' \in D_\times^m$. En effet,

$$\|\hat{\mathbf{z}}'\|_\infty = \|\hat{\mathbf{s}}\mathbf{r}_j + \hat{\mathbf{y}}\|_\infty \leq \|\hat{\mathbf{s}}\mathbf{r}_j\|_\infty + \|\hat{\mathbf{y}}\|_\infty \leq \sqrt{n}\log n + mn^{1.5}\log n.$$

On remarque qu'après $q_{_H}$ requêtes à l'oracle aléatoire, la probabilité d'avoir une collision sur H_L est plus petite que $q_{_H}/|D_{s,c}|$. Par conséquent, la probabilité que $\mathcal{F}$ retourne une signature forgée qui est une solution pour le problème $\mathsf{Col}(h, D_\times)$ est au moins égale à $\delta - q_{_H}/|D_{s,c}|$. Dans la suite, on montre que la probabilité d'obtenir une solution pour le problème $\mathsf{Col}(h, D_\times)$ lorsque $\mathbf{r}_j$ est une réponse à une requête pour l'oracle aléatoire envoyée par $\mathcal{F}$ est plus petite que $\delta - q_{_H}/|D_{s,c}|$. Cette nouvelle valeur sera une borne inférieur de la probabilité de succès pour résoudre le problème $\mathsf{Col}(h, D_\times)$.

Maintenant on s'intéresse au cas où $\mathbf{r}_j$ est une réponse à une requête pour l'oracle aléatoire envoyée par $\mathcal{F}$. Dans ce cas, on enregistre la sortie de $\mathcal{F}$ qui correspond au couple message/signature $(\mu, \hat{\mathbf{z}}, \mathbf{r}_j)$. Puis on génère des nouvelles valeurs $\mathbf{r}'_j, \cdots, \mathbf{r}'_{q_{_H}} \leftarrow D_{s,c}$. Ensuite, on relance l'algorithme $\mathcal{B}$ une autre fois mais avec $(h, \hat{\mathbf{s}}, b_s, b_f, \mathbf{r}_1, \cdots, \mathbf{r}_{j-1}, \mathbf{r}'_j, \cdots, \mathbf{r}'_{q_{_H}})$. D'après le Lemme B.1, on a que la probabilité que $\mathbf{r}'_j \neq \mathbf{r}_j$ et l'attaquant $\mathcal{F}$ utilise la réponse de l'oracle aléatoire $\mathbf{r}'_j$ et la requête associée à cette valeur dans sa forge est au moins égale à

$$\left(\delta - \frac{1}{|D_{s,c}|}\right)\left(\frac{\delta - 1/|D_{s,c}|}{q_{_H}} - \frac{1}{|D_{s,c}|}\right),$$

Par conséquent, avec la probabilité ci-dessus, $\mathcal{F}$ retourne une nouvelle signature $(\hat{\mathbf{z}}', \mathbf{e}')$ du message μ avec $\mathbf{e}' = \mathbf{r}'_j$ et $h(\hat{\mathbf{z}}') - \mathbf{Se}' = h(\hat{\mathbf{z}}) - \mathbf{Se}$. En utilisant

les propriétés homomorphes de h et puisqu'on connait la clé secrète $\hat{\mathbf{s}}$ telle que $h(\hat{\mathbf{s}}) = \mathbf{S}$, on obtient l'équation suivante

$$h(\hat{\mathbf{z}} - \hat{\mathbf{s}}\mathbf{e}) = h(\hat{\mathbf{z}}' - \hat{\mathbf{s}}\mathbf{e}'). \tag{B.3}$$

Si $\hat{\mathbf{z}} - \hat{\mathbf{s}}\mathbf{e} = \hat{\mathbf{z}}' - \hat{\mathbf{s}}\mathbf{e}'$, alors on obtient une collision sur h. On va utiliser le fait que le schéma de signature satisfait la propriété de témoin indistinguable, pour montrer que $\hat{\mathbf{z}} - \hat{\mathbf{s}}\mathbf{e} = \hat{\mathbf{z}}' - \hat{\mathbf{s}}\mathbf{e}'$. D'après le Lemme 1.7, il existe une autre clé secrète possible $\hat{\mathbf{s}}' \in D_{s,c}^m$ telle que $h(\hat{\mathbf{s}}) = h(\hat{\mathbf{s}}')$ et par le Théorème 1.5, on sait que la probabilité de deviner quelle clé secrète a été utilisée par le signataire est au plus égale à $1/2 + n^{-\omega(1)}$. Par conséquent, on a que la clé secrète utilisée est différente de $\hat{\mathbf{s}}$ avec une probabilité $1/2 - n^{-\omega(1)}$.

On va montrer que si $\hat{\mathbf{z}} - \hat{\mathbf{s}}\mathbf{e} = \hat{\mathbf{z}}' - \hat{\mathbf{s}}\mathbf{e}'$, alors pour tout $\hat{\mathbf{s}}' \neq \hat{\mathbf{s}}$, on aura $\hat{\mathbf{z}} - \hat{\mathbf{s}}'\mathbf{e} \neq \hat{\mathbf{z}}' - \hat{\mathbf{s}}'\mathbf{e}'$. Ceci se démontre par l'absurde. Supposons que

$$\hat{\mathbf{z}} - \hat{\mathbf{s}}\mathbf{e} = \hat{\mathbf{z}}' - \hat{\mathbf{s}}\mathbf{e}' \text{ et } \hat{\mathbf{z}} - \hat{\mathbf{s}}'\mathbf{e} = \hat{\mathbf{z}}' - \hat{\mathbf{s}}'\mathbf{e}',$$

en soustrayant les deux équations, on obtient que $(\hat{\mathbf{s}} - \hat{\mathbf{s}}')(\mathbf{e} - \mathbf{e}') = 0$. Puisqu'on fait les opérations dans l'anneau $\mathcal{D} = \mathbb{Z}_p[x]/\langle x^n + 1\rangle$ qui n'est pas intègre, alors on ne peut pas en déduire que soit $(\hat{\mathbf{s}} - \hat{\mathbf{s}}') = 0$ ou bien $(\mathbf{e} - \mathbf{e}') = 0$. En revanche, on peut faire la remarque suivante : puisque toutes les coordonnées de $\hat{\mathbf{s}}, \hat{\mathbf{s}}', \mathbf{e}$ et $\mathbf{e}'$ sont dans $\{-1, 0, 1\}$, alors $\hat{\mathbf{s}} - \hat{\mathbf{s}}'$ et $\mathbf{e} - \mathbf{e}'$ ont des coordonnées dans $\{-2, -1, 0, 1, 2\}$. Donc lorsqu'on fait la multiplication de ces polynômes dans l'anneau $\mathbb{Z}[x]/\langle x^n + 1\rangle$, les valeurs absolues des coefficients du produit sont inférieures à $4n$. Ceci implique que lorsqu'on passe modulo $p = \Theta(n^4)$ aucun coefficient du produit ne sera réduit parce que $4n < p/2$. Par conséquent, si le produit $(\hat{\mathbf{s}} - \hat{\mathbf{s}}')(\mathbf{e} - \mathbf{e}')$ est égal à 0 dans l'anneau $\mathbb{Z}_p[x]/\langle x^n + 1\rangle$ alors il sera égal à 0 dans l'anneau $\mathbb{Z}[x]/\langle x^n + 1\rangle$. En outre, on a sait que le polynôme $x^n + 1$ est irréductible dans $\mathbb{Z}[x]$, donc $\mathbb{Z}[x]/\langle x^n + 1\rangle$ est un anneau intègre et par conséquent soit $(\hat{\mathbf{s}} - \hat{\mathbf{s}}') = 0$ ou bien $(\mathbf{e} - \mathbf{e}') = 0$. Puisque on sait que $\mathbf{e} \neq \mathbf{e}'$, alors forcément $\hat{\mathbf{s}} = \hat{\mathbf{s}}'$ et on obtient une contradiction. Donc $\hat{\mathbf{z}} - \hat{\mathbf{s}}\mathbf{e} \neq \hat{\mathbf{z}}' - \hat{\mathbf{s}}\mathbf{e}'$ avec probabilité au moins $1/2 - n^{-\omega(1)}$ et on a une collision sur h. Pour finir la preuve il suffit de montrer que $\|\hat{\mathbf{z}} - \hat{\mathbf{s}}\mathbf{e}\|_\infty$ et $\|\hat{\mathbf{z}}' - \hat{\mathbf{s}}\mathbf{e}'\|_\infty$ sont dans l'ensemble $D_\times^m$. On sait que les réponses $\mathbf{r}_i$ de l'oracle H_L sont choisies d'une manière uniformément aléatoire dans $D_{s,c}$, donc en utilisant le Lemme 2.5, on obtient que pour tout $\mathbf{r}_i$ la norme $\|\hat{\mathbf{s}}\mathbf{r}_i\|_\infty$ est au plus égale à $\sqrt{n}\log n$ avec une probabilité de $1 - n^{-\omega(1)}$. Ainsi, puisque $\hat{\mathbf{z}}, \hat{\mathbf{z}}' \in D_z^m$ alors on a $\|\hat{\mathbf{z}} - \hat{\mathbf{s}}\mathbf{e}\|_\infty$, $\|\hat{\mathbf{z}}' - \hat{\mathbf{s}}\mathbf{e}'\|_\infty \leq mn^{1.5}\log n$.

Conclusion : L'algorithme $\mathcal{B}$ trouve une solution pour le problème de collision lié à h avec une probabilité au moins égale à

$$\left(\frac{1}{2} - n^{-\omega(1)}\right)\left(\delta - \frac{1}{|D_{s,c}|}\right)\left(\frac{\delta - 1/|D_{s,c}|}{q_H} - \frac{1}{|D_{s,c}|}\right).$$

Bibliographie

[AABN02] Michel Abdalla, Jee Hea An, Mihir Bellare, and Chanathip Namprempre. From identification to signatures via the Fiat-Shamir transform : Minimizing assumptions for security and forward-security. In *Advances in Cryptology-EUROCRYPT 2002*, pages 418–433. Springer, 2002.

[AD97] Miklós Ajtai and Cynthia Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 284–293. ACM, 1997.

[Ajt96] M. Ajtai. Generating hard instances of lattice problems (extended abstract). In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, STOC '96, pages 99–108, New York, NY, USA, 1996. ACM.

[Ajt98] Miklós Ajtai. The shortest vector problem in ℓ_2 is NP-hard for randomized reductions. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 10–19. ACM, 1998.

[Ajt99] Miklós Ajtai. Generating hard instances of the short basis problem. In *Automata, Languages and Programming*, pages 1–9. Springer, 1999.

[AMBB+13] Carlos Aguilar Melchor, Slim Bettaieb, Xavier Boyen, Laurent Fousse, and Philippe Gaborit. Adapting Lyubashevsky's signature schemes to the ring signature setting. In *Progress in Cryptology–AFRICACRYPT 2013*, pages 1–25. Springer, 2013.

[AOS02] Masayuki Abe, Miyako Ohkubo, and Koutarou Suzuki. 1-out-of-n signatures from a variety of keys. *Advances in Cryptology-ASIACRYPT 2002*, pages 639–645, 2002.

[AP11] Joël Alwen and Chris Peikert. Generating shorter bases for hard random lattices. *Theory of Computing Systems*, 48(3) :535–553, 2011.

[BCC88] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences*, 37(2) :156–189, 1988.

[BK10] Zvika Brakerski and Yael Tauman Kalai. A framework for efficient signatures, ring signatures and identity based encryption in the standard model. *IACR Cryptology ePrint Archive*, 2010 :86, 2010.

[BKM06] Adam Bender, Jonathan Katz, and Ruggero Morselli. Ring signatures : Stronger definitions, and constructions without random oracles. In *Proceedings of TCC 2006, volume 3876 of LNCS*, pages 60–79. Springer-Verlag, 2006.

[BN06] M. Bellare and G. Neven. Multi-signatures in the plain public-key model and a general forking lemma. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 390–399. ACM, 2006.

[Boy07] Xavier Boyen. Mesh signatures. In *Advances in Cryptology - EUROCRYPT 2007, 26th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Barcelona, Spain, May 20-24, 2007, Proceedings*, pages 210–227, 2007.

[Boy10] Xavier Boyen. Lattice mixing and vanishing trapdoors – a framework for fully secure short signatures and more. In *Public Key Cryptography—PKC 2010*, volume 6056 of *Lecture Notes in Computer Science*, pages 499–517. Berlin : Springer-Verlag, 2010. Available at http://www.cs.stanford.edu/~xb/pkc10b/.

[BP02] Mihir Bellare and Adriana Palacio. GQ and schnorr identification schemes : Proofs of security against impersonation under active and concurrent attacks. In *Advances in Cryptology-CRYPTO 2002*, pages 162–177. Springer, 2002.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical : A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM conference on Computer and communications security*, pages 62–73. ACM, 1993.

[Bre02] Emmanuel Bresson. *Protocoles Cryptographiques pour l'Authentification et l'Anonymat dans les Groupes*. PhD thesis, École polytechnique, 2002.

[BS99] Johannes Blömer and Jean-Pierre Seifert. On the complexity of computing short linearly independent vectors and short bases in a lattice. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 711–720. ACM, 1999.

[BS13] Slim Bettaieb and Julien Schrek. Improved lattice-based threshold ring signature scheme. In *Post-Quantum Cryptography*, pages 34–51. Springer, 2013.

[BSS02] Emmanuel Bresson, Jacques Stern, and Michael Szydlo. Threshold ring signatures and applications to ad-hoc groups. In *Advances in Cryptology-Crypto 2002*, pages 465–480. Springer, 2002.

[CHK09] David Cash, Dennis Hofheinz, and Eike Kiltz. How to delegate a lattice basis. *IACR Cryptology ePrint Archive*, 2009 :351, 2009.

[CHKP10] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In *Advances in Cryptology–EUROCRYPT 2010*, pages 523–552. Springer, 2010.

[CLRS10a] Pierre-Louis Cayrel, Richard Lindner, Markus Rückert, and Rosemberg Silva. Improved zero-knowledge identification with lattices. In *Provable Security*, pages 1–17. Springer, 2010.

[CLRS10b] Pierre-Louis Cayrel, Richard Lindner, Markus Rückert, and Rosemberg Silva. A lattice-based threshold ring signature scheme. In *Progress in Cryptology–LATINCRYPT 2010*, pages 255–272. Springer, 2010.

[CN97] Jin-Yi Cai and Ajay Nerurkar. An improved worst-case to average-case connection for lattice problems. In *ANNUAL SYMPOSIUM ON FOUNDATIONS OF COMPUTER SCIENCE*, volume 38, pages 468–479. Citeseer, 1997.

[CVA11] Pierre-Louis Cayrel, Pascal Véron, and Sidi Mohamed El Yousfi Alaoui. A zero-knowledge identification scheme based on the q-ary syndrome decoding problem. In *Selected Areas in Cryptography*, pages 171–186. Springer, 2011.

[CVH91] David Chaum and Eugène Van Heyst. Group signatures. In *Advances in Cryptology-EUROCRYPT'91*, pages 257–265. Springer, 1991.

[DB13] Léo Ducas-Binda. *Signatures Fondées sur les Réseaux Euclidiens : Attaques, Analyses et Optimisations*. PhD thesis, École Normale Supérieure, 2013. http://www.di.ens.fr/~ducas/Thesis/thesis.pdf.

[DH76] Whitfield Diffie and Martin E Hellman. New directions in cryptography. *Information Theory, IEEE Transactions on*, 22(6) :644–654, 1976.

[DKNS04] Yevgeniy Dodis, Aggelos Kiayias, Antonio Nicolosi, and Victor Shoup. Anonymous identification in ad hoc groups. In *Advances in Cryptology-EUROCRYPT 2004*, pages 609–626. Springer, 2004.

[FFS88] Uriel Feige, Amos Fiat, and Adi Shamir. Zero-knowledge proofs of identity. *Journal of cryptology*, 1(2) :77–94, 1988.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself : Practical solutions to identification and signature problems. In *CRYPTO*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1986.

[FS90] Uriel Feige and Adi Shamir. Witness indistinguishable and witness hiding protocols. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 416–426. ACM, 1990.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, volume 9, pages 169–178, 2009.

[GGH96] Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Collision-free hashing from lattice problems. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 3, pages 236–241, 1996.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of computer and system sciences*, 28(2) :270–299, 1984.

[GMR85] 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*, pages 291–304. ACM, 1985.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2) :281–308, 1988.

[Gol07] O. Goldreich. *Foundations of Cryptography : Volume 1, Basic Tools*. Foundations of Cryptography. Cambridge University Press, 2007.

[GPV08] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the 40th annual ACM symposium on Theory of computing*, pages 197–206. ACM, 2008.

[GSV98] Oded Goldreich, Amit Sahai, and Salil Vadhan. Honest-verifier statistical zero-knowledge equals general statistical zero-knowledge. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 399–408. ACM, 1998.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4) :1364–1396, 1999.

[HS03] J. Herranz and G. Sáez. Forking lemmas for ring signature schemes. *Progress in Cryptology-INDOCRYPT 2003*, pages 266–279, 2003.

[Kan83] Ravi Kannan. Improved algorithms for integer programming and related lattice problems. In *Proceedings of the fifteenth annual ACM symposium on Theory of computing*, pages 193–206. ACM, 1983.

[Kho04] Subhash Khot. Hardness of approximating the shortest vector problem in lattices. In *Foundations of Computer Science, 2004. Proceedings. 45th Annual IEEE Symposium on*, pages 126–135. IEEE, 2004.

[KTX08] Akinori Kawachi, Keisuke Tanaka, and Keita Xagawa. Concurrently secure identification schemes based on the worst-case hardness of

lattice problems. In *Advances in Cryptology-ASIACRYPT 2008*, pages 372–389. Springer, 2008.

[LHA+12] Arjen K Lenstra, James P Hughes, Maxime Augier, Joppe W Bos, Thorsten Kleinjung, and Christophe Wachter. Ron was wrong, whit is right. *IACR Cryptology ePrint Archive*, 2012 :64, 2012.

[LM06] Vadim Lyubashevsky and Daniele Micciancio. Generalized compact knapsacks are collision resistant. In *Automata, Languages and Programming*, pages 144–155. Springer, 2006.

[Lyu08a] Vadim Lyubashevsky. Lattice-based identification schemes secure under active attacks. In *Public Key Cryptography–PKC 2008*, pages 162–179. Springer, 2008.

[Lyu08b] Vadim Lyubashevsky. *Towards practical lattice-based cryptography*. PhD thesis, University of California, San Diego, 2008. Available at http://www.di.ens.fr/~lyubash/papers/dissertation_singlespaced.pdf.

[Lyu09] Vadim Lyubashevsky. Fiat-shamir with aborts : Applications to lattice and factoring-based signatures. In *Advances in Cryptology–ASIACRYPT 2009*, pages 598–616. Springer, 2009.

[Lyu12] Vadim Lyubashevsky. Lattice signatures without trapdoors. In *Advances in Cryptology–EUROCRYPT 2012*, pages 738–755. Springer, 2012.

[MG02] Daniele Micciancio and Shafi Goldwasser. *Complexity of lattice problems : a cryptographic perspective*, volume 671. Springer, 2002.

[Mic04] Daniele Micciancio. Almost perfect lattices, the covering radius problem, and applications to ajtai's connection factor. *SIAM Journal on Computing*, 34(1) :118–169, 2004.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices : Simpler, tighter, faster, smaller. In *Advances in Cryptology–EUROCRYPT 2012*, pages 700–718. Springer, 2012.

[MP13] Daniele Micciancio and Chris Peikert. Hardness of SIS and LWE with small parameters. In *Advances in Cryptology–CRYPTO 2013*, pages 21–39. Springer, 2013.

[MR07] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on Gaussian measures. *SIAM Journal on Computing*, 37(1) :267–302, 2007.

[MV03] Daniele Micciancio and Salil P Vadhan. Statistical zero-knowledge proofs with efficient provers : Lattice problems and more. In *Advances in Cryptology-CRYPTO 2003*, pages 282–298. Springer, 2003.

[MV10a] Daniele Micciancio and Panagiotis Voulgaris. A deterministic single exponential time algorithm for most lattice problems based on voronoi cell computations. In *Proceedings of the 42nd ACM symposium on Theory of computing*, pages 351–358. ACM, 2010.

[MV10b] Daniele Micciancio and Panagiotis Voulgaris. Faster exponential time algorithms for the shortest vector problem. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1468–1480. Society for Industrial and Applied Mathematics, 2010.

[MVOV96] Alfred J Menezes, Paul C Van Oorschot, and Scott A Vanstone. *Handbook of applied cryptography*. CRC press, 1996.

[Pei09] Chris Peikert. Bonsai Trees (or, arboriculture in Lattice-Based Cryptography). *IACR Cryptology ePrint Archive*, 2009 :359, 2009.

[Poi95] David Pointcheval. A new identification scheme based on the perceptrons problem. In *Advances in Cryptology-EUROCRYPT'95*, pages 319–328. Springer, 1995.

[PW11] Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. *SIAM Journal on Computing*, 40(6) :1803–1844, 2011.

[Reg08] Oded Regev. Lecture notes on lattices in computer science, 2004. 28, 2008. Available at http://www.cims.nyu.edu/~regev/teaching/lattices_fall_2004.

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6) :34, 2009.

[RS10] Markus Rückert and Michael Schneider. Estimating the Security of Lattice-based Cryptosystems. *IACR Cryptology ePrint Archive*, 2010 :137, 2010.

[RSA78] Ronald L Rivest, Adi Shamir, and Len Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2) :120–126, 1978.

[RST01] Ronald L Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In *Advances in Cryptology-ASIACRYPT 2001*, pages 552–565. Springer, 2001.

[Rüc10] Markus Rückert. Strongly unforgeable signatures and hierarchical identity-based signatures from lattices without random oracles. In *Post-Quantum Cryptography*, pages 182–200. Springer, 2010.

[Sho97] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM journal on computing*, 26(5) :1484–1509, 1997.

[SSH11] Koichi Sakumoto, Taizo Shirai, and Harunaga Hiwatari. Public-key identification schemes based on multivariate quadratic polynomials. In *Advances in Cryptology–CRYPTO 2011*, pages 706–723. Springer, 2011.

[SSTX09] Damien Stehlé, Ron Steinfeld, Keisuke Tanaka, and Keita Xagawa. Efficient public key encryption based on ideal lattices. In *Advances in Cryptology–ASIACRYPT 2009*, pages 617–635. Springer, 2009.

[SW07] Hovav Shacham and Brent Waters. Efficient ring signatures without random oracles. In *Public Key Cryptography–PKC 2007*, pages 166–180. Springer, 2007.

[vEB81] Peter van Emde-Boas. *Another NP-complete partition problem and the complexity of computing short vectors in a lattice*. Department, Univ., 1981.

[WS11] Jin Wang and Bo Sun. Ring signature schemes from lattice basis delegation. In *Proceedings of the 13th international conference on Information and communications security*, pages 15–28. Springer-Verlag, 2011.

[Xag10] Keita Xagawa. *Cryptography with Lattices*. PhD thesis, Department of Mathematical and Computing Sciences, Tokyo Institute of Technology, 2010. Available at http://xagawa.net/pdf/2010Thesis.pdf.