Université de Caen Basse-Normandie

U.F.R. de sciences

École doctorale SIMEM

THÈSE

présentée par

M. **Fabien Laguillaumie**

et soutenue

le mercredi 29 juin 2005

en vue de l'obtention du

DOCTORAT de l'UNIVERSITÉ de CAEN BASSE-NORMANDIE

Spécialité : mathématiques et leurs applications

(Arrêté du 25 avril 2002)

Signatures à vérification contrôlée basées sur des applications bilinéaires : conception et analyse de sécurité

MEMBRES du JURY

Éric Reyssat, professeur à l'Université de Caen Basse-Normandie (directeur de thèse)
Antoine Joux, HDR, DGA, professeur associé à l'Université de Versailles (rapporteur)
David Pointcheval, HDR, chargé de recherche CNRS, École normale supérieure (rapporteur)
Thierry Berger, professeur à l'Université de Limoges
Marc Girault, HDR, expert émérite à France Télécom Recherche et Développement
Jacques Stern, professeur à l'École normale supérieure
Brigitte Vallée, directrice de recherche CNRS, Université de Caen Basse-Normandie (invitée)

Remerciements

L'écriture de la page des remerciements offre l'opportunité de remercier toutes les personnes qui ont contribué à faire d'une thèse ce qu'elle est (et annonce la fin de celle-ci) : c'est avec plaisir que je la saisis.

En premier lieu, mes remerciements vont à mon directeur de thèse, Éric Reyssat, qui a accepté de m'encadrer dans un domaine qui n'est pas le sien. Patient et d'une grande disponibilité, Éric a assimilé toutes les facettes de la cryptographie qui se sont présentées à lui, et a pu m'apporter des points de vue et des conseils pertinents. Il m'a par ailleurs laissé une grande liberté durant toute ma thèse, et je l'en remercie.

Je remercie Marc Girault pour ses commentaires éclairés sur mon travail, sur ses extensions possibles et pour le temps (précieux) qu'il a passé à relire et corriger ce manuscrit. Il a été présent tout au long de ma thèse, et en particulier dans ses derniers moments : il a fait plus pour moi qu'il ne devait et je suis très heureux qu'il fasse partie de mon jury. Côtoyer Marc a été, autant du point de vue humain que scientifique, véritablement enrichissant.

Cette thèse ne serait évidemment pas ce qu'elle est sans Damien Vergnaud, et sans cette collaboration qui s'est créée dès le début de nos thèses. Nous avons partagé (outre un bureau magnifique) un grand nombre d'expériences liées à la vie de jeunes chercheurs : stress, conférences, moments de joie, deadlines, stress,... Ses grandes qualités scientifiques ont été un moteur pour notre recherche. Je le remercie amicalement.

Brigitte Vallée a toujours été disponible pour répondre à mes questions existentielles et me montrer le droit chemin. Je lui suis grandement reconnaissant pour le temps (professionnel et extra professionnel) qu'elle m'a consacré, et souhaite lui témoigner toute mon amitié. Merci pour tout Brigitte.

Je remercie vivement Antoine Joux et David Pointcheval d'avoir accepté de rapporter cette thèse. Je remercie également David pour les contacts que nous avons eus pendant ma thèse, et pour l'intérêt qu'il a porté à mon travail. Ses résultats ont été une grande source d'inspiration.

Jacques Stern me fait l'honneur de participer à mon jury : je lui en suis reconnaissant et l'en remercie.

Merci également à Thierry Berger d'être membre de mon jury. C'est aussi grâce à Thierry que je fais une thèse à Caen. Je garde un excellent souvenir de mes années universitaires à Limoges, et je profite de cette occasion pour remercier les nombreuses personnes du LACO qui m'ont donné envie de continuer en thèse.

Une grande partie de ma thèse s'est déroulée à ~~FTR&D~~ France Télécom Recherche et Développement dans l'URD ~~DTL/SSR~~ MAPS/NSS. Le cadre de travail était particulièrement agréable, et je remercie toute l'URD pour la bonne humeur permanente.

Jean-François Misarsky a encadré mon stage de DEA, puis il m'a accompagné en thèse CIFRE comme responsable industriel. Je le remercie pour ses conseils et son aide pour me lancer dans le grand bain.

Un grand merci à Jacques Traoré pour l'intérêt qu'il a porté à mes travaux, pour ses remarques et ses encouragements.

Je remercie spécialement l'équipe *old school* des jeunes qui montent : Benjamin (la hot - very hot - line debian), Seb (le cryptologue averti) et Vince (spécialiste des urgences à toute heure). Merci aussi à la *new generation* (c'est bientôt fini Elvis !), et bon courage aux nouveaux thésards !

Un merci particulier à Hervé pour son soutien et son amitié.

Merci à Aline pour tous les bons moments.

J'ai eu la chance de pouvoir évoluer à la fac pendant ma période en entreprise. Ce fut l'occasion de nombreuses discussions autour d'un café ou dans les couloirs, avec les matheux, comme avec les informaticiens.

Je tiens plus particulièrement à remercier Ali Akhavi pour ses conseils et son aide. Merci à Ayoub, Céline, Guillaume, et à ceux de l'autre côté de la passerelle : Ben (qui écrit des articles immoraux [67]), Jean-Marie et Philippe.

La cryptographie à Caen s'est notamment concrétisée par le séminaire Crypto et les deux colloques CAEN'X, avec $X \in \{03, 05\}$. Je remercie le triangle cryptographique caennais réuni à l'occasion de ces manifestations, ainsi que les invités et les spectateurs.

Une thèse est l'occasion de nombreuses rencontres humaines et scientifiques. Je remercie donc toutes les personnes croisées lors de conférences ou d'écoles de jeunes chercheurs avec lesquelles j'ai gardé d'excellents rapports.

Je remercie Mathieu Ciet, François Koeune, et Jean-Jacques Quisquater du Crypto Group de l'UCL pour notre collaboration. Je tiens à remercier Jean-Jacques pour son accueil à Louvain-la-Neuve lors de mes deux séjours, ainsi que l'ensemble des membres du groupe, et en particulier Benoît Libert pour les différentes rencontres et discussions. Je remercie aussi Pascal Paillier, rencontré à Amsterdam à l'occasion d'Eurocrypt 2002, qui a par la suite accepté les invitations caennaises. Les connaissances de Pascal en cryptographie sont impressionnantes, et c'est un véritable plaisir de travailler avec lui, dans une atmosphère à la fois sérieuse et empreinte de bonne humeur.

L'année de DEA précédant ma thèse a été particulièrement mouvementée : c'est grâce en particulier à Greg, Lolo, Manue, Pat, Seb, Tom. C'est toujours un plaisir de se retrouver autour d'un verre.

Merci à tous mes amis de plus longue date : Anthony (très longue date !), Baba, Ben, Marie, Mehdi, Nani (merci beaucoup), Thibault, Steph, Sylvain, ainsi que tous ceux que j'oublie.

Caen est une ville agréable pour y préparer une thèse, et je remercie en vrac le Lux (le Café des Images aussi), le Brouillon (merci Caro !), le Bouchon, le Panta, le BBC, Tapes à cent balles,...

Enfin, je remercie mes parents pour leur soutien et leur aide constants. Ils ont toujours été présents et attentifs : je leur dois beaucoup. Je remercie mon frère Pierre et Laura pour les relectures sérieuses et précises des articles en anglais (malgré le délai souvent extrèmement court...), mais aussi pour leurs conseils avisés.

Avant de se lancer dans la lecture de ce manuscrit, un avertissement nécessaire, à propos d'Alice et Bob :

Vous le savez que tous les personnages
de ce récit sont fictifs et que toute
ressemblance, etc., etc.
Hmm ?
Alors

S.-A.

Table des matières

Remerciements **1**

Notations **11**

Table des figures **15**

Introduction générale **17**

Partie I Généralités **21**

Chapitre 1
Éléments de Cryptologie **23**

1.1 Les outils 24
1.1.1 Complexité 24
Machine de Turing 24
Description 24
Notions essentielles 24
1.1.2 Outils cryptographiques 25
Fonctions de hachage 26
Indistinguabilité de variables aléatoires 27
1.2 Cryptographie à clé publique 27
1.2.1 Chiffrement 27
Modèle de sécurité 27
1.2.2 Authentification 30

1.2.3 Signature numérique 30
Modèle de sécurité 30
1.2.4 Les preuves de sécurité 32
1.2.4.1 Preuves et réductions 32
1.2.4.2 Les différents modèles 32
Modèle standard 32
Modèle de l'oracle aléatoire 33
Modèle générique 33
Fonctions de hachage 33
1.2.5 Preuves interactives à divulgation nulle de connaissance 34
1.2.5.1 Définitions 34
1.2.5.2 Exemples 36
Égalité de deux logarithmes discrets 36
Protocole d'authentification de Schnorr 36
1.2.5.3 L'heuristique Fiat-Shamir 37
1.3 Hypothèses calculatoires 37
1.3.1 RSA 38
1.3.2 Logarithme discret 39
1.3.3 Diffie-Hellman et ses variantes 40
1.3.3.1 Introduction 40
1.3.3.2 Diffie-Hellman calculatoire 41
1.3.3.3 Diffie-Hellman décisionnel 42
1.3.3.4 Gap Diffie-Hellman 42
1.3.4 Diffie-Hellman et applications bilinéaires 43
1.3.4.1 Diffie-Hellman bilinéaire 44
1.3.4.2 Des variantes exotiques 46
ℓ-SDH 46
CCA 48

Chapitre 2
Des mathématiques pour la cryptologie 49

2.1 Les entiers 49
2.1.1 Factorisation 50
2.1.2 RSA 50
2.1.3 Logarithme discret 52
Les algorithmes pour calculer des logarithmes discrets 53
2.2 Courbes elliptiques 53

2.2.1 Définitions . . . 53
2.2.2 Comptage des points d'une courbe elliptique . . . 55
2.2.3 Logarithme discret sur une courbe elliptique . . . 56
2.2.4 Remarques . . . 57
2.2.4.1 Suites de divisibilité elliptiques . . . 57
2.2.4.2 Autres applications cryptographiques . . . 58
2.3 Applications bilinéaires . . . 58
2.3.1 Exemples . . . 59
2.3.1.1 Couplages de Weil et Tate . . . 59
Définitions . . . 60
Propriétés . . . 61
Algorithme de Miller . . . 61
2.3.1.2 Exponentiation discrète . . . 62

Chapitre 3
Retour sur le chiffrement et la signature

3.1 Chiffrement . . . 63
3.1.1 Sécurité sémantique - IND-CCA . . . 64
3.1.1.1 Définition . . . 64
3.1.2 Indistinguabilité des clés : Key-Privacy . . . 65
3.1.2.1 Définition . . . 65
3.2 Signature . . . 66
3.2.1 Modèle formel . . . 66
3.2.2 Diverses signatures utiles par la suite . . . 67
3.2.2.1 Signatures courtes . . . 68
3.2.2.2 Signatures d'anneau . . . 69
3.3 Remarques . . . 72
3.3.1 Enregistrement des clés des utilisateurs . . . 72
3.3.2 Preuves de sécurité . . . 72

Partie II Signatures électroniques : variantes 75

Chapitre 1
Préambule

1.1 Introduction . . . 77
1.2 L'astuce xyz . . . 78
1.2.1 Le problème xyz-DDH . . . 78
1.2.2 De nouveaux problèmes . . . 82

Chapitre 2
Signatures indéniables

2.1 Introduction . . . 85
2.1.1 Définition . . . 87
Une nouvelle propriété : la conversion temporelle . . . 87
2.1.2 Modèle de sécurité . . . 88
2.1.2.1 Consistance des vérifications . . . 88
2.1.2.2 Inforgeabilité . . . 89
2.1.2.3 Anonymat/Invisibilité . . . 89
2.1.2.4 Non-transférabilité . . . 91
2.2 CUSBM : nouvelles signatures indéniables convertibles . . . 92
2.3 TSCUSBM : signatures indéniables convertibles dans le temps. . . . 92
2.4 Preuves de sécurité . . . 93
2.4.1 Inforgeabilité . . . 94
2.4.2 Anonymat . . . 98
2.5 Signatures indéniables courtes dans le modèle standard . . . 100
2.5.1 Preuve d'égalité de logarithme discret . . . 101
2.5.2 USWRO : description . . . 103
2.5.3 Preuves de sécurité . . . 103
2.5.3.1 Anonymat . . . 103
2.5.3.2 Inforgeabilité . . . 108
2.6 Conclusion . . . 108

Chapitre 3
Signatures dirigées

3.1 Introduction . . . 111
3.1.1 Définition . . . 112
Idée générale . . . 112
Définition formelle . . . 113
3.1.2 Modèle de sécurité . . . 114
3.1.2.1 Consistance et significativité . . . 114
3.1.2.2 Inforgeabilité . . . 115
3.1.2.3 Invisibilité . . . 115
Invisibilité faible . . . 116
Invisibilité forte . . . 116
3.1.3 Une nouvelle hypothèse calculatoire . . . 117
Diffie-Hellman carré flexible . . . 117
3.2 Nouveau schéma . . . 118
3.2.1 Description . . . 118
3.2.2 Preuve de sécurité . . . 119
3.2.2.1 Inforgeabilité . . . 119
3.2.2.2 Invisibilité . . . 125
Invisibilité faible . . . 125
Invisibilité forte . . . 127
3.2.3 Corollaire : une variante probabiliste de Boneh-Boyen . . . 128
3.3 Conclusion . . . 129

Chapitre 4
Signatures à vérificateur(s) désigné(s)

4.1 Définition . . . 132
Applications . . . 132
4.1.1 Définition . . . 133
4.1.2 Modèle de sécurité . . . 134
4.1.2.1 Consistance . . . 134
4.1.2.2 Inforgeabilité . . . 134
4.1.2.3 Ambiguïté du signataire . . . 135
4.1.2.4 Anonymat du signataire . . . 136
Définition . . . 136
Chiffrement IND-CCA2 implique anonymat . . . 137
4.2 Au singulier : signatures à vérificateur désigné . . . 138

4.2.1 État de l'art . . . 138
Remarques sur la sécurité de certains de ces schémas . . . 139
4.2.1.1 Signatures à vérificateur désigné universelles . . . 140
4.2.1.2 MAC et DVS . . . 141
Définition . . . 142
Inforgeabilité . . . 143
Indistinguabilité . . . 143
4.2.1.3 Forge universelle sur le schéma de DVS de Sehkar . . . 144
Description du schéma . . . 144
Forge universelle par une attaque sans message . . . 144
4.2.2 DVSBMH . . . 145
4.2.2.1 Description . . . 145
4.2.2.2 Avantages de cette construction . . . 145
4.2.2.3 Preuve de sécurité dans le cas général . . . 147
Correctness et *source hiding* . . . 147
Inforgeabilité . . . 147
Anonymat . . . 149
4.2.2.4 Preuve de sécurité dans le cas où $\mathbb{G}_0 = \mathbb{G}_1$ . . . 150
Inforgeabilité . . . 150
Anonymat . . . 150
4.2.3 Comparaison des schémas . . . 153
Sécurité fixée . . . 153
Sécurité industrielle . . . 154
4.3 Au pluriel : signatures à vérificateurs désignés . . . 154
4.3.1 Construction générique à partir de signatures d'anneaux . . . 155
4.3.1.1 Description . . . 155
4.3.1.2 À propos de MDVS . . . 155
4.3.2 Un schéma efficace dans le cas $n = 2$ . . . 156
4.3.2.1 Description . . . 157
4.3.2.2 Preuves de sécurité . . . 157
Consistance et *source hiding* . . . 157
Inforgeabilité . . . 157
Anonymat . . . 160
4.3.3 SMDVS : anonymat sans chiffrement . . . 162
4.3.3.1 Description . . . 162
4.3.3.2 Sécurité . . . 162
Consistance et source hiding . . . 162

Inforgeabilité 163
Anonymat 164
4.3.3.3 Remarque 166
4.3.4 Conclusion 166

Chapitre 5
Conclusion

Bibliographie **171**

Notations

$\mathbb{N}$	ensemble des entiers naturels
$\mathbb{Z}$	ensemble des entiers relatifs
$\mathbb{Z}/N\mathbb{Z}$	anneau des entiers modulo N
$\mathbb{R}$	ensemble des nombres réels
A^*	ensemble des éléments inversibles (pour la multiplication) de l'anneau A
$\#E$	cardinal de l'ensemble E
$\bigcap \bigcup$ et $\bigwedge \bigvee$	opérateurs ensemblistes et logiques "et" et "ou"
$[\![a, b]\!]$	intervalle des entiers entre a et b
p	un nombre premier
$\{0,1\}^*$	ensemble des suites de bits de longueur quelconque
$a \vert\vert b$	concaténation des suites de bits a et b
$\vert x \vert$	taille en bits de x
$x \in_R E$ ou $x \overset{R}{\longleftarrow} E$	l'élément x est tiré au hasard dans l'ensemble fini E selon une distribution uniforme
$T_{\mathbb{G}}$	temps d'évaluation d'une opération dans le groupe $\mathbb{G}$
$T_{\mathcal{O}}$	temps de réponse de l'oracle $\mathcal{O}$
T_P	temps d'évaluation d'un couplage

$a \leftarrow b$	la valeur de b est affectée à a
$x \leftarrow \mathcal{A}(\ldots)$	l'algorithme $\mathcal{A}$ produit l'élément x en sortie
$\Pr[R_1; \ldots; R_n : E]$	probabilité de l'événement E après les exécutions successives des événements $R_1, \ldots, R_n$
DVS	abréviation de “signatures à vérificateur désigné”
$IZKP(\ldots)$	preuve interactive à divulgation nulle de connaissance
$NIZKP(\ldots)$	preuve non interactive à divulgation nulle de connaissance
$DVP(\ldots)$	preuve à vérificateur désigné
EF-CMA	*existential unforgeability against a chosen message attack*
Ano-CMA	*anonymity against a chosen message attack*
PSI-CMA	*privacy of signer's identity against a chosen message attack*
SH-CMA	*source hiding against a chosen message attack*
weak-Inv-CMA	*weak invisibility against a chosen message attack*
strong-Inv-CMA	*strong invisibility against a chosen message attack*
IND-CCA2	*indistinguishability against an adaptative chosen ciphertext attack*

Nous utilisons la description suivante des expériences aléatoires modélisant une attaque sur un problème algorithmique :

$\boxed{\text{Experiment } \mathbf{Exp}^{\mathsf{problème}-r}_{\mathsf{Gen},\mathcal{A}}(k)}$

E_1

$\vdots$

E_n

Sortie

Cette expérience aléatoire décrit une attaque de l'attaquant $\mathcal{A}$ sur le générateur Gen d'instances de problème par rapport à un paramètre de sécurité k et dépendant éventuellement d'un bit r si l'attaquant est du type décisionnel. Des événements $E_1, \ldots, E_n$ se succèdent jusqu'à la fin de l'expérience qui se termine par une sortie Sortie $\in \{0, 1\}$ finale.

L'analogue pour des attaques sur des cryptosystèmes est décrite ici :

$$\boxed{\text{Experiment } \mathbf{Exp}^{\mathsf{attaque}-r}_{\mathsf{Schéma},\mathcal{A}}(k)}$$

E_1

$\vdots$

E_n

Sortie

Cette expérience aléatoire décrit une attaque de l'attaquant $\mathcal{A}$ sur le schéma Schéma dans une attaque du type attaque.

Suivant la sortie de l'expérience, on définit un *succès* (si l'attaquant est du type calculatoire) et un *avantage* (si l'attaquant est du type décisionnel) de la façon suivante :

$$\mathbf{Succ}^{\mathsf{zz}}_{\mathsf{XX},\mathcal{YY}}(k) = \Pr\left[\mathbf{Exp}^{\mathsf{zz}}_{\mathsf{XX},\mathcal{YY}}(k) = 1\right]$$
$$\mathbf{Adv}^{\mathsf{zz}}_{\mathsf{XX},\mathcal{YY}}(k) = \left|\Pr\left[\mathbf{Exp}^{\mathsf{zz}-1}_{\mathsf{XX},\mathcal{YY}}(k) = 1\right] - \Pr\left[\mathbf{Exp}^{\mathsf{zz}-0}_{\mathsf{XX},\mathcal{YY}}(k) = 1\right]\right|$$

La probabilité est considérée sur les aléas internes des algorithmes intervenant dans les événements $E_1, \ldots, E_n$ et sur les différents tirages aléatoires.

Table des figures

1.1 Preuve d'égalité de 2 logarithmes discrets . . . 36
1.2 Protocole d'authentification en 3 passes . . . 37
1.3 Protocole d'authentification de Schnorr . . . 37
1.4 Signatures de Schnorr . . . 38
1.5 Échange de clé Diffie-Hellman . . . 40
1.6 Échange de clé Diffie-Hellman tripartite . . . 44

2.1 Chiffrement RSA "textbook" . . . 51
2.2 Signature RSA FDH . . . 52
2.3 Algorithme de Miller pour le couplage de Tate . . . 62

3.1 Relations entre les notions de sécurité du chiffrement . . . 65
3.2 Signatures BLS . . . 69
3.3 Signatures Boneh-Boyen . . . 70
3.4 Signatures d'anneau de Boneh, Gentry, Lynn et Shacham . . . 71

2.1 CUSBM : signature . . . 93
2.2 CUSBM : confirmation, démenti . . . 93
2.3 CUSBM : conversion, vérification . . . 94
2.4 TSCUSBM : signature . . . 95
2.5 TSCUSBM : confirmation, démenti . . . 95
2.6 TSCUSBM : conversion, vérification . . . 96
2.7 Preuve non transférable d'égalité de logarithme discret . . . 102
2.8 Preuve non transférable d'inégalité de logarithme discret . . . 102
2.9 USWRO : description . . . 104

3.1 DSBM : signature . . . 120
3.2 DSBM : vérification . . . 121
3.3 DSBM : confirmation, démenti . . . 121
3.4 DSBM : conversion individuelle et vérification . . . 122
3.5 DSBM : reçu de conversion totale et vérification . . . 122
3.6 Signatures Boneh-Boyen probabilistes . . . 129

4.1 DVSBM . . . 141
4.2 DVSBMH . . . 146
4.3 Comparaison entre quelques DVS à sécurité égale . . . 154
4.4 Comparaison entre quelques DVS (sécurité industrielle) . . . 154
4.5 MDVS . . . 156

4.6 B2DVS 158
4.7 SMDVS 163

Introduction générale

La cryptologie est une science appliquée, largement implantée dans notre environnement immédiat. Souvent définie comme la *science du secret* [233], la cryptologie se divise en deux parties. La première, la cryptographie, consiste en l'élaboration et l'étude de systèmes cryptographiques. Le côté obscur, la cryptanalyse, constitue la deuxième partie, résolument tournée vers l'attaque des précédents systèmes. Ce mémoire s'inscrit plutôt dans la première des branches de la cryptologie, bien que la cryptanalyse nous ait permis d'écarter plusieurs essais de conception de schémas de signatures.

L'objectif de la cryptographie est d'assurer, dans un environnement non sûr, des notions d'intégrité, d'authenticité ou de confidentialité. De plus, des contraintes, souvent physiques, liées aux développements des (nano-)technologies de communications, imposent aux algorithmes cryptographiques d'être légers, rapides et multi-fonctions.

Pour réaliser ces objectifs, la cryptographie moderne s'est dotée d'un cadre différent de celui instauré depuis son antique apparition. Tout d'abord, les bases théoriques ont été posées par Shannon au milieu des années 1940, et réunies dans [225], après qu'il ait développé sa théorie de l'information dans [224]. Cependant, les grands principes de la cryptographie, toujours admis à l'heure actuelle, avaient été énoncés par Kerckhoffs en 1883 [134][1]. Par la suite, Diffie et Hellman, en inventant en 1976 le concept de cryptographie à clé publique [76], ont suggéré que l'échange d'un secret pour assurer la sécurité des communications n'était pas nécessaire. Dorénavant, Alice et Bob n'auront plus à se mettre d'accord sur un secret, en l'échangeant sur un canal sécurisé, ou de main à main. Chacun possèdera deux clés : l'une est publique et diffusée ; l'autre, intimement liée à la précédente, est gardée secrète. Le premier algorithme de chiffrement à clé publique a été publié par Rivest, Shamir et Adleman en 1978 [208] et a marqué le début de l'ère de la cryptographie à clé publique. La cryptographie a clé secrète n'a pas pour autant disparu, les deux cohabitent pour obtenir les meilleurs compromis en matière de sécurité et de performance.

L'authentification est l'une des primitives cryptographiques les plus importantes, à travers l'identification et la signature. Elle est l'objet de ce mémoire. Des variantes de la signature électronique seront présentées, variantes motivées par les applications électroniques de ces signatures. Les exigences requises pour les signatures numériques, qui se voudraient l'équivalent des signatures manuscriptes, sont en fait beaucoup plus nombreuses et beaucoup plus subtiles que celles de ces dernières. De plus, la variété des applications étant particulièrement large, de nombreuses propriétés supplémentaires sont souhaitées, en particulier des propriétés d'anonymat, souvent nécessaires dans les applications sensibles comme le vote électronique, ou le paiement électronique par exemple.

Nous allons décrire des schémas de signatures dotés de propriétés particulières et fournir la preuve de leur sécurité. Contrairement aux schémas proposés au tout début de la cryptographie à

[1]"le système doit être matériellement, sinon mathématiquement, indécryptable". "Il faut qu'il n'exige pas le secret et qu'il puisse sans inconvénient tomber entre les mains de l'ennemi".

clé publique, des arguments informels de sécurité et la résistance à la cryptanalyse ne constituent plus une preuve suffisante de sécurité. L'étude de la sécurité se fait dorénavant dans un contexte formel, après avoir exhibé précisément les buts et les moyens de l'attaquant.

L'introduction, en 2000, du couplage de Weil en cryptographie [127] a substantiellement élargi l'horizon de la conception d'algorithmes. Elle a permis en particulier de répondre a un problème ouvert vieux d'une vingtaine d'années : le chiffrement basé sur l'identité [35]. Cet outil, issu de la géométrie algébrique, servira de brique à l'essentiel des schémas présentés dans ce mémoire.

Contributions et organisation du mémoire

C'est dans ce contexte que vont être présentés les résultats de ce mémoire. Ils constituent une étude des schémas de signatures "spéciales", possédant des propriétés d'anonymat, en particulier pour le signataire. La première partie présente des outils et des résultats généraux de cryptographie à clé publique, et des mathématiques sous-jacentes.

La deuxième partie de ce mémoire regroupe les résultats obtenus durant cette thèse. Le point de départ a été le concept de signatures à vérificateur désigné, qui, pour des raisons de cohérence dans la présentation, sera développé dans le dernier chapitre. En effet, l'ancêtre commun à la majorité des signatures "spéciales" est le schéma de signatures indéniables de Chaum et van Antwerpen présenté à Crypto'89 [57]. C'est l'une des premières fois qu'il est question du contrôle de la vérification d'une signature, et du fait que la vérification universelle n'est pas une propriété nécessairement souhaitable. Ce sont des signatures qui ne peuvent convaincre de l'authenticité d'un message que lors d'une interaction avec le signataire. La conviction acquise à l'issue de cette interaction ne peut être transmise à une autre personne. Nous proposons une variante convertible en signatures universellement vérifiables. Nous définissons ensuite une notion nouvelle, la conversion temporelle. Les schémas obtenus possèdent la propriété d'anonymat du signataire et leur sécurité repose sur des variantes bilinéaires du problème Diffie-Hellman. Nous proposons aussi un schéma de signatures indéniables (sans propriété additionnelle) dont la sécurité peut être prouvée dans le modèle standard. Les résultats ont fait l'objet d'une publication à RSA-CT'05 [148], et d'une soumission [147].

Le chapitre suivant est consacré à une extension des signatures indéniables : les signatures dirigées, introduites en 1993 par Lim et Lee [157, 158]. La vérification d'une signature peut se faire grâce à une interaction avec une autre personne que le signataire. Nous apportons dans ce chapitre une étude formelle de ce concept, ainsi que le premier schéma de signatures dirigées universellement convertibles. Ces résultats ont été soumis [142].

Enfin, le dernier chapitre est consacré aux signatures à vérificateur désigné, qui sont en fait des versions non interactives des preuves à vérificateur désigné, introduites par Jakobsson, Sako et Impagliazzo en 1996. Ces preuves ont été proposées pour améliorer les protocoles de confirmation et de démenti d'une signature dans les signatures indéniables. Elles peuvent naturellement être rendues non interactives en utilisant le paradigme de Fiat-Shamir. Les signatures qui en découlent (qu'on pourrait appeler authentification répudiable non interactive) ont la propriété de ne convaincre de leur (in)validité que le vérificateur désigné : en effet ce dernier, comme le signataire, peut produire une signature. La propriété de non-répudiation n'est plus vérifiée pour ces signatures. Dans un premier temps, nous proposons un schéma de signatures courtes à vérificateur désigné, basé sur n'importe quelle application bilinéaire, y compris l'exponentiation discrète. Nous étendons ensuite le concept pour permettre la désignation de plusieurs vérificateurs. Nous proposons une propriété d'anonymat du signataire, qui ne peut être obtenue dans le schéma initial en présence de plusieurs vérificateurs sans chiffrer la signature. Nous présentons donc un autre schéma qui protège l'anonymat du signataire, au prix d'une taille de signature linéaire en le nombre de vérificateurs. Ces résultats ont été publiés à SCN'04 [145] et à ICICS'04 [144], ou soumis [146].

Première partie

Généralités

1

Éléments de Cryptologie

Sommaire

1.1 Les outils 24
1.1.1 Complexité 24
1.1.2 Outils cryptographiques 25
1.2 Cryptographie à clé publique 27
1.2.1 Chiffrement 27
1.2.2 Authentification 30
1.2.3 Signature numérique 30
1.2.4 Les preuves de sécurité 32
1.2.5 Preuves interactives à divulgation nulle de connaissance 34
1.3 Hypothèses calculatoires 37
1.3.1 RSA 38
1.3.2 Logarithme discret 39
1.3.3 Diffie-Hellman et ses variantes 40
1.3.4 Diffie-Hellman et applications bilinéaires 43

Ce chapitre est un chapitre introductif qui présente les briques de base de la cryptographie. Nous y décrivons tout d'abord des objets issus de la théorie de la complexité, nécessaire pour fixer les notations et préciser les objets manipulés (algorithme, réduction, complexité,. . .). Nous présentons ensuite les outils cryptographiques élémentaires, qui sont le fondement de la théorie.

Une fois ce cadre d'étude précisé, nous aborderons les problématiques spécifiques et naturelles de la cryptographie à clé publique et les diverses manières d'y répondre. Les signatures constitueront le cœur de ce mémoire, mais nous esquisserons aussi les autres techniques cryptographiques importantes.

Finalement, nous expliciterons un certain nombre d'hypothèses algorithmiques sur lesquelles reposent de nombreux cryptosystèmes et qui sont indispensables pour s'assurer de la sécurité de ceux-ci. La sécurité des systèmes cryptographiques est, en effet, reliée à la difficulté de résoudre des problèmes algorithmiques. Cependant, pour une grande majorité d'entre eux, cette difficulté est complètement heuristique et non prouvée. C'est le cas de la factorisation du produit de deux grands entiers, ou du calcul d'un logarithme discret sur une courbe elliptique. Il est donc nécessaire de faire des hypothèses (les plus raisonnables possibles) consistant à supposer la difficulté de certains problèmes.

1.1 Les outils

1.1.1 Complexité

La théorie de la complexité est nécessaire en cryptographie pour appréhender la difficulté des problèmes sur lesquels reposent les protocoles développés. La notion d'algorithme et de leur complexité, celle de machine de Turing, sont des notions fondamentales pour étudier la sécurité des cryptosystèmes de manière formelle. Aussi, dans cette partie, nous donnons un certain nombre de définitions issues de la théorie de la complexité, et les plaçons dans le contexte de la cryptographie.

Machine de Turing Les machines de Turing ont été inventées par Turing à la fin des années 30 [237]. Ce modèle de calcul permet d'atteindre l'un des buts de la théorie de la complexité, à savoir la classification des problèmes algorithmiques selon leur difficulté. Cet objet abstrait permet de décrire de façon formelle la notion "d'opération élémentaire", et de formaliser la notion de "fonction calculable". La théorie montre que ce modèle permet en particulier de simuler de façon raisonnable un ordinateur. En ce qui nous concerne, elles nous permettront de voir un *algorithme* et les entités cryptographiques[2] comme des machines de Turing.

Description Une *machine de Turing à k bandes (ou rubans)* est un quintuplet $\mathcal{M} = (\Sigma, Q, \delta, q_0, F)$ où

- Σ est un ensemble fini de symboles, l'alphabet de la machine, contenant un symbole particulier B,
- Q est un ensemble non vide d'états,
- q_0 est l'état initial,
- F est un sous-ensemble de Q appelé ensemble des états finaux,
- $\delta : Q \times \Sigma^k \longrightarrow \Sigma^{k-1} \times Q \times \{\texttt{gauche},\texttt{droite}\}$ est la fonction de transition de la machine.

Les bandes sont constituées d'une infinité de cellules juxtaposées linéairement. La première bande est appelée bande d'entrée de la machine, et les suivantes bandes de travail. La machine peut les lire, et y écrire. Sur chaque cellule ne peut être inscrit qu'au plus 1 symbole de Σ. À chaque instant, un nombre fini de cases contient un symbole de Σ différent de B. Les bandes sont pourvues d'une tête de lecture/écriture qui se déplace d'une case à une autre. Pour fonctionner, la bande d'entrée reçoit une suite finie de symboles de $\Sigma \setminus \{B\}$, et la tête de lecture est placée sur le premier symbole de cette suite. La machine est alors à l'état initial et exécute le programme.

Notions essentielles

- Si une donnée x est inscrite initialement sur le ruban de la machine de Turing, la *complexité* $T_M(x)$ d'un calcul est le nombre de pas élémentaires nécessaires pour passer de l'état initial à l'arrêt (éventuel) de la machine.

- La complexité d'un calcul est dite *polynomiale* si elle est majorée par un polynôme en la taille n des données initiales. On parle de machine de Turing fonctionnant en temps polynomial et on la note $PTM(n)$.

[2] Alice et Bob seront aussi des machines de Turing

– Les machines de Turing que nous considérerons par la suite possèdent également une source de bits aléatoires, modélisée par un ruban aléatoire, contenant une suite infinie de bits tirés au hasard. Si l'espérance de la complexité de la machine par rapport à son ruban aléatoire est polynomiale, elle est appelée machine de Turing probabiliste fonctionnant en temps polynomial, et notée $PPTM(n)$.

– Une machine de Turing *interactive* possède en outre un ruban de données initiales et un ruban de communication en lecture/écriture.

– Un *protocole interactif* est une paire de machines de Turing interactives partageant les bandes supplémentaires de données initiales et de communication.

– Un dernier type de machine nous sera utile par la suite : ce sont les *machines à oracle*. Cette machine a accès à une puissance de calcul arbitraire pour un certain type de problème et obtient une réponse à une instance de ce problème en un temps constant. Si $\mathcal{A}$ est une telle machine, ayant accès à un oracle $\mathcal{O}$, nous la noterons $\mathcal{A}^{\mathcal{O}}$.

Ainsi, dans la suite, un *algorithme* sera une machine de Turing ; un *algorithme efficace* et un *adversaire* seront des machines de Turing interactives probabilistes fonctionnant en temps polynomial.

1.1.2 Outils cryptographiques

La notion fondamentale de la cryptographie est celle de *fonction à sens unique*. Informellement, une fonction à sens unique est une fonction pour laquelle il existe un algorithme efficace qui la calcule, alors qu'un algorithme efficace ne saura l'inverser qu'avec une probabilité négligeable. Un bon nombre de résultats théoriques en cryptographie repose sur l'existence de telles fonctions : générateurs pseudo-aléatoires, chiffrement symétrique,.... Leur existence implique, par ailleurs, que $\mathcal{P} \neq \mathcal{NP}$.

Les définitions suivantes introduisent de façon formelle les objets sous-jacents à la construction et à l'analyse des divers cryptosystèmes exposés par la suite. La notion de fonction négligeable est nécessaire pour quantifier les définitions de ces fonctions cryptographiques. Une fonction négligeable est une fonction qui est asymptotiquement plus petite que l'inverse de n'importe quel polynôme.

Définition 1 (Fonction négligeable) *Une fonction $\nu : \mathbb{N} \longrightarrow \mathbb{R}^+$ est dite* négligeable *si pour tout entier $c \geq 0$, il existe un entier k_c tel que $\nu(k) < k^{-c}$, pour tout $k \geq k_c$.*

On peut alors donner une définition formelle d'une fonction à sens unique (*one-way function*), et d'une collection de fonctions à sens unique.

Définition 2 (Fonction à sens unique) *Une fonction $f : \{0,1\}^* \longrightarrow \{0,1\}^*$ est dite* à sens unique *si*

1. *il existe un algorithme efficace qui, prenant x en entrée, renvoie $f(x)$;*
2. *pour tout algorithme efficace $\mathcal{A}$, il existe une fonction négligeable $\nu_{\mathcal{A}}$ telle que, pour k suffisamment grand,*

$$\Pr\left[x \xleftarrow{R} \{0,1\}^k; y \leftarrow f(x); x' \leftarrow \mathcal{A}(1^k, y) : f(x') = f(x)\right] \leq \nu_{\mathcal{A}}(k).$$

Définition 3 *Une collection de fonctions* $F = \{f_k : \{0,1\}^k \longrightarrow \{0,1\}^k, k \in \mathbb{N}\}$ *est dite* à sens unique *si*

1. *il existe un algorithme efficace qui, prenant* k *et* x *en entrée, renvoie* $f_k(x)$ *;*
2. *pour tout algorithme efficace* $\mathcal{A}$*, il existe une fonction négligeable* $\nu_{\mathcal{A}}$ *telle que, pour* k *suffisamment grand,*

$$\Pr\left[x \xleftarrow{R} \{0,1\}^k; y \leftarrow f_k(x); x' \leftarrow \mathcal{A}(1^k, y) : f(x') = f(x)\right] \leq \nu_{\mathcal{A}}(k).$$

Une autre des notions élémentaires fondamentales est la notion de trappe. Une fonction à sens unique est à trappe (*trapdoor function*) si la connaissance d'une donnée particulière permet de la calculer efficacement dans le sens difficile.

Définition 4 (Fonction à trappe) *Une fonction* $f : \{0,1\}^* \longrightarrow \{0,1\}^*$ *est dite* à trappe *si elle est à sens unique, et s'il existe un algorithme efficace* $\mathcal{B}$ *et un entier* $c \geq 0$ *tel que, pour tout* $k \in \mathbb{N}$*, il existe une suite de bits* $t_k \in \{0,1\}^*$ *avec* $|t_k| \leq k^c$ *et pour tout* $x \in \{0,1\}^k$*,*

$$\mathcal{B}(1^k, f(x), t_k) = x' \text{ et } f(x) = f(x').$$

Définition 5 *Une collection de fonctions à trappe* $F = \{f_k : \{0,1\}^k \longrightarrow \{0,1\}^k, k \in \mathbb{N}\}$ *est une collection de fonctions à sens unique pour laquelle il existe un algorithme efficace* $\mathcal{B}$ *et un entier* $c \geq 0$ *tel que, pour tout* $k \in \mathbb{N}$*, il existe une suite de bits* $t_k \in \{0,1\}^*$ *avec* $|t_k| \leq k^c$ *et pour tout* $x \in \{0,1\}^k$*,*

$$\mathcal{B}(1^k, f_k(x), t_k) = x' \text{ et } f_k(x) = f_k(x').$$

Beaucoup de résultats théoriques sont donc liés à l'existence de fonctions à sens unique, ou de fonctions à trappe. Par exemple, on peut montrer aussi que s'il existe une collection de permutations à sens unique, alors il existe une collection de générateurs pseudo-aléatoires, qui entraîne l'existence d'une collection de fonctions pseudo-aléatoires, ces objets étant aussi au centre de la cryptographie. Des exemples de fonctions admises comme étant à sens unique seront donnés dans la section 1.3.

Fonctions de hachage Les fonctions de hachage sont un outil essentiel en cryptographie. Elles permettent essentiellement de produire une empreinte de longueur fixe d'une suite quelconque de bits, et par exemple de plonger un message dans un codage particulier (en un point d'une courbe elliptique, en un élément du groupe multiplicatif d'un corps fini,...). Elles sont cruciales dans l'étude de la sécurité des cryptosystèmes. Nous reviendrons sur ces fonctions dans la partie 1.2.4.2. Intuitivement, si $h : \{0,1\}^* \longrightarrow \{0,1\}^\ell$ est une fonction de hachage, elle doit vérifier les propriétés suivantes :

- résistance à la préimage, ce qui signifie qu'elle est à sens unique,
- résistance à la seconde préimage : étant donné un message $m \in \{0,1\}^*$, il est calculatoirement difficile de trouver un message $m' \neq m$ tel que $h(m) = h(m')$:
- résistance aux collisions :il est calculatoirement difficile de trouver deux éléments différents m et m' dans $\{0,1\}^*$ tels que $h(m) = h(m')$

Indistinguabilité de variables aléatoires Finalement, nous quantifions ici la notion de ressemblance de deux familles de variables aléatoires, à laquelle on fait appel pour comparer diverses distributions. Soit $X = \{X(m)\}_{m\in L}$ et $Y = \{Y(m)\}_{m\in L}$ deux familles de variables aléatoires paramétrées par les mots d'un langage $L \subset \{0,1\}^*$. Pour tout $m \in L$, $X(m)$ et $Y(m)$ sont donc des variables aléatoires à valeurs dans $\{0,1\}^*$. Les familles X et Y sont dites *statistiquement indistinguables* s'il existe une fonction négligeable ν telle que

$$\forall m \in L \qquad \sum_{\alpha\in\{0,1\}^*} \left| \Pr[X(m) = \alpha] - \Pr[Y(m) = \alpha] \right| < \nu(|m|).$$

Elles sont dites *parfaitement indistinguables* si la somme ci dessus est nulle pour tout $m \in L$. Finalement, elles sont dites *polynomialement (calculatoirement) indistinguables* si pour tout algorithme efficace $\mathcal{D}$ qui renvoie 0 ou 1, il existe une fonction negligeable ν telle que

$$\left| \Pr[\alpha \xleftarrow{R} X(m) : \mathcal{D}(\alpha) = 1] - \Pr[\alpha \xleftarrow{R} Y(m) : \mathcal{D}(\alpha) = 1] \right| < \nu(|m|)$$

1.2 Cryptographie à clé publique

Dans cette partie, nous détaillons les briques essentielles fournies par la cryptographie à clé publique pour atteindre ses objectifs de confidentialité, d'intégrité, d'authentification ou de non-répudiation.

1.2.1 Chiffrement

Le chiffrement, défini dans le paragraphe suivant, est un procédé destiné à assurer la confidentialité. Son but est donc de rendre un message inintelligible à toute personne qui n'en est pas le destinataire légitime.

Modèle de sécurité Dans un schéma de chiffrement à clé publique, chaque utilisateur possède un couple de clés, dont l'une est rendue publique, et l'autre gardée secrète. L'opération de chiffrement se fait grâce à la clé publique de la personne à qui l'on veut envoyer le message chiffré, et l'opération de déchiffrement se fait grâce à la clé secrète.

Définition 6 (Schéma de chiffrement à clé publique) *Soit k un entier. Un* schéma de chiffrement *de paramètre de sécurité k est composé des algorithmes suivants :*
- génération des paramètres publics Setup *: algorithme probabiliste prenant en entrée un entier k et renvoyant* les paramètres publics *;*
- génération de clés KeyGen *: algorithme probabiliste prenant en entrée un entier k, des paramètres publics et produisant une* paire de clés publique et secrète (pk, sk) *;*
- chiffrement $\mathcal{E}$ *: algorithme déterministe ou probabiliste prenant en entrée un entier k, des paramètres publics, une clé publique pk et un message $m \in \{0,1\}^*$, et renvoyant un* chiffré *c de m ;*
- déchiffrement $\mathcal{D}$ *: algorithme déterministe prenant en entrée un entier k, des paramètres publics, une clé secrète sk, et un chiffré c, et renvoyant le message m dont c est le chiffré.*

Intuitivement, on attend d'un tel schéma qu'il vérifie au moins les propriétés suivantes (en fait, on en demande beaucoup plus) :

- un message correctement chiffré sera correctement déchiffré,

– retrouver le message m en connaissant juste son chiffré doit être difficile sans la clé secréte de déchiffrement.

Plus précisément, pour étudier la sécurité d'un tel algorithme de chiffrement (et de tout schéma cryptographique), il importe de définir les attaques contre lesquelles on veut se prémunir, ainsi que les moyens dont dispose l'attaquant pour les mettre en œuvre. Dans le cas du chiffrement, on distingue 2 types d'attaquant : le premier va tenter de retrouver la totalité de l'information cachée dans un chiffré. Son objectif est alors :

– le *cassage total* ; en d'autres termes, il retrouve la clé secrète de l'utilisateur qu'il attaque,

– le *sens unique* de la fonction de chiffrement ; l'attaquant veut pouvoir déchiffrer à volonté, c'est-à-dire retrouver le texte clair correspondant à un chiffré donné, sans pour autant connaître la clé secrète de l'utilisateur cible.

Le deuxième type d'attaquant est moins ambitieux : son but est seulement d'obtenir un peu d'information sur un texte clair. Dans ce cas, l'attaquant procède en 2 étapes. Dans la première phase, il reçoit la clé publique de l'utilisateur qu'il veut attaquer, et, à l'issue de cette phase (et après un certain nombre d'opérations de sa part), propose un test à un challenger. Ce dernier lui présente alors un chiffré challenge c. L'information qu'en retire l'attaquant peut lui permettre de distinguer un message chiffré (*indistinguabilité du chiffrement*), ou bien de construire une relation entre des chiffrés (*non-malléabilité*). Plus précisément,

– la *sécurité sémantique* (ou *indistinguabilité*, *sécurité polynomiale*) du chiffrement a été introduite par Goldwasser et Micali en 1982 [111,112] dans leur article à propos du chiffrement probabiliste. En effet, il est nécessaire que le chiffrement soit probabiliste pour obtenir cette propriété. On peut la décrire ainsi : à l'issue de sa première phase, l'attaquant propose deux messages m_0 et m_1 (de même longueur) au challenger, qui chiffre l'un d'eux, et présente le chiffré c à l'attaquant. Ce dernier doit tenter de retrouver lequel des deux messages a été chiffré (avec une probabilité significativement meilleure que la probabilité de réussir en donnant une réponse au hasard, soit $1/2$). Si le schéma de chiffrement est sémantiquement sûr, l'attaquant n'apprend aucune information sur le message,

– la *non-malléabilité* du chiffrement a été introduite par Dolev, Dwork et Naor en 1991 [79,80], puis étudiée dans [16] et [26] par Bellare *et al.*. Intuitivement, l'attaquant exploite de l'information contenue dans un chiffré, sans connaître celle-ci. À l'issue de sa première phase, l'attaquant propose un ensemble de textes clairs de même longueur, décrit. Le challenge c est le chiffré de m qui est l'un des messages de cet espace. L'attaquant, dans sa deuxième phase, va tenter de produire une relation k-aire $\mathcal{R}$ et un ensemble de $k-1$ chiffrés valides $C = \{c_1, \ldots, c_k\}$, avec $c \notin C$ tels que $\mathcal{R}(m, \mathcal{D}(sk, c_1), \ldots, \mathcal{D}(sk, c_k))$ soit vraie.

Pour tenter de briser les propriétés précédentes, l'attaquant peut se lancer dans :

– une *attaque à textes clairs choisis* (*Chosen Plaintext Attack* - CPA) durant laquelle l'attaquant a seulement à sa disposition le chiffré de textes clairs de son choix (ce qui est toujours possible puisqu'il est en possession de la clé publique qu'il veut attaquer),

– une *attaque à messages chiffrés choisis* (*Chosen Ciphertext Attack* - CCA1) introduite par Naor et Yung en 1990 [180] (sous le terme de *lunch-time attack*) durant laquelle l'attaquant obtient le déchiffrement séquentiel de messages de son choix, avant de proposer son test au challenger. Dans sa deuxième phase, l'attaquant n'a plus l'accès à l'oracle de déchiffrement.

– une *attaque à messages chiffrés choisis adaptative* (CCA2) introduite par Rackoff et Simon [204], et pendant laquelle l'attaquant peut obtenir le déchiffrement de chiffrés de son choix, et ce même après avoir reçu le chiffré challenge (bien évidemment, il ne peut pas obtenir le déchiffré du chiffré challenge). Cette attaque est donc adaptative en ce sens que l'attaquant peut tenir compte de la connaissance du challenge pour interroger, dans sa deuxième phase, l'oracle de déchiffrement.

Une technique classique pour créer un algorithme de chiffrement est d'utiliser une fonction à trappe. Les problèmes essentiels liés à leur utilisation telle quelle sont leur caractère déterministe, ainsi que le fait que la connaissance de $f(x)$ ne cache pas nécessairement toute l'information à propos de x. La fonction RSA, par exemple, préserve le symbole de Jacobi de x. De même, l'exponentiation discrète ne cache pas le bit de poids faible de x. Pour obtenir un chiffrement probabiliste, la primitive de base reste la fonction à trappe, mais plus précisément ce sont les notions de prédicats *hard-core* pour des fonctions à trappe, introduites par Blum et Micali dans [31], et de prédicats à trappe, introduite par Goldwasser et Micali dans [111, 112], qui sont fondamentales.

Un prédicat *hard-core* d'une fonction à sens unique $f : \{0,1\}^* \longrightarrow \{0,1\}^*$ est défini comme étant un prédicat booléen $B : \{0,1\}^* \longrightarrow \{0,1\}$ tel que

– il existe un algorithme efficace A tel que $\forall\, x \in \{0,1\}^*\ A(x) = B(x)$ et
– pour tout algorithme efficace G, il existe une fonction négligeable ν_G telle que pour k suffisamment grand,

$$\Pr\left[x \xleftarrow{R} \{0,1\}^k; y \leftarrow f(x); b \leftarrow B(x); b' \leftarrow G(1^k, y) : b' = b\right] \leq \frac{1}{2} + \nu_G(k).$$

Intuitivement, un prédicat sur x est *hard-core* s'il est difficile étant donné $f(x)$ de le trouver autrement qu'en le donnant au hasard.

Un prédicat *à sens unique* est une fonction booléenne $B : \{0,1\}^* \longrightarrow \{0,1\}$ pour laquelle :
– il existe un algorithme efficace qui, prenant en entrée $v \in \{0,1\}$ et $k \in \mathbb{N}$, tire aléatoirement un x tel que $B(x) = v$ et $|x| \leq k$;
– pour tout algorithme efficace G, il existe une fonction négligeable ν_G telle que pour k suffisamment grand,

$$\Pr\left[x \xleftarrow{R} \{0,1\}^k; y \leftarrow f(x); b \leftarrow B(x); b' \leftarrow G(1^k, y) : b' = b\right] \leq \frac{1}{2} + \nu_G(k).$$

Un prédicat à trappe est un prédicat à sens unique pour lequel il existe, pour tout $k \in \mathbb{N}$, une information t_k de taille majorée par un polynôme en k, dont la connaissance permet le calcul en temps polynomial de $B(x)$ pour tout x vérifiant $|x| \leq k$.

Ces deux prédicats peuvent être simplement utilisés pour chiffrer un seul bit, et on peut étendre cette utilisation pour chiffrer des messages de taille quelconque, comme l'ont montré Blum et Goldwasser dans [30]. Leur schéma obtient les propriétés de sécurité sémantique. Cependant, cette contruction est peu efficace (comme la plupart des constructions théoriques) et n'est donc pas satisfaisante en pratique. Aussi, en 1994, Bellare et Rogaway proposent un schéma nommé OAEP (*Optimal Asymmetric Encryption Padding*) qui consiste en une transformation utilisant des fonctions de hachage, du message en une suite de bits de longueur fixe. C'est cette suite de bits qui passera dans la fonction de chiffrement. La sécurité de ce schéma nécessite l'hypothèse idéale que les fonctions de hachage se comportent comme des fonctions aléatoires, mais il constitue un des schémas les plus performants ayant toutes les propriétés de sécurité voulues. L'étude de la sécurité se trouve dans [228] et [93], la variante de Boneh [33] étant la plus efficace.

1.2.2 Authentification

L'authentification consiste pour une entité à convaincre un interlocuteur de son identité. En général, celui qui cherche à prouver son identité est appelé le *prouveur*, et s'adresse à un *vérificateur*. Une authentification est donc un protocole (interactif) à l'issue duquel un prouveur honnête a convaincu un vérificateur qu'il est bien celui qu'il prétend être : on parle aussi d'*identification*. Prouver son identité consiste essentiellement à prouver que l'on connaît un secret. Les mécanismes les plus utilisés à l'heure actuelle sont ceux qui consistent à le dévoiler (comme les mots de passe). Nous allons cependant avoir pour les techniques d'authentification des exigences de sécurité pour lesquelles la divulgation du secret n'est pas acceptable. En particulier, on souhaite d'un tel protocole qu'un vérificateur ne puisse pas se faire passer pour un prouveur qui s'est déjà authentifié auprès de lui, et que même après voir participé à plusieurs exécutions du protocole (même dévié) il ne puisse pas usurper l'identité du prouveur.

Ainsi, un attaquant contre un schéma d'authentification peut être *passif* : il ne fait qu'observer une série d'exécutions du protocole d'authentification entre un prouveur et un vérificateur, puis il va ensuite tenter de se faire passer pour ce prouveur face à un vérificateur honnête. S'il est *actif*, l'attaquant va éventuellement participer à des exécutions du protocole en prenant par exemple la place du vérificateur, pour essayer (en choisissant les informations échangées durant le protocole) de capturer de l'information qui lui permettra par la suite de se faire passer pour le prouveur. La notion de sécurité admise pour les schémas d'authentification (introduite par Feige, Fiat et Shamir dans [86]) est donc cette notion de *résistance à l'impersonnification face à des attaques actives et concurrentes*.

Les mécanismes "ancestraux" comme l'utilisation du mot de passe ne rentrent pas dans cette catégorie, bien qu'ils soient largement répandus. Les méthodes qui vont attirer notre attention sont celles basées sur des preuves à divulgation nulle de connaissance (voir la partie 1.2.5). Parmi ces schémas, on peut citer le schéma d'authentification Fiat et Shamir [88], de Schnorr [215], de Guillou-Quisquater [119] (la sécurité de ces deux derniers schémas face à des attaques actives et concurrentes a été récemment prouvée par Bellare et Palacio [21]) ou GPS [103, 104].

1.2.3 Signature numérique

La signature numérique authentifie, non un individu, mais un message. Elle apporte une preuve que le message provient d'un utilisateur donné, et qu'il n'a pas été modifié. La signature se calcule grâce à une clé secrète, et la vérification grâce à la clé publique associée.

Modèle de sécurité La définition formelle d'un schéma de signature est la suivante :

Définition 7 (Schéma de signatures numériques) *Soit k un entier. Un* schéma de signatures numériques *de paramètre de sécurité k est composé des algorithmes suivants :*

- génération des paramètres publics Setup *: algorithme probabiliste prenant en entrée un entier k et renvoyant* les paramètres publics *;*
- génération de clés KeyGen *: algorithme probabiliste prenant en entrée un entier k, des paramètres publics et produisant une* paire de clés publique et secrète (pk, sk) *;*

– signature *Sign : un algorithme déterministe ou probabiliste prenant en entrée un entier* k*, des paramètres publics, une clé secrète* sk *et un message* $m \in \{0,1\}^*$*, et renvoyant une* signature σ *de* m *;*
– vérification *Verify : algorithme déterministe prenant en entrée un entier* k*, des paramètres publics, une clé publique* pk*, un message* $m \in \{0,1\}^*$ *et une suite de bits* σ *et vérifiant que* σ *est une signature valide du message* m*. Il répond* 1 *si c'est le cas, 0 sinon.*

Un tel schéma de signature sera satisfaisant s'il vérifie les deux propriétés suivantes :

– toute signature correctement formée doit être acceptée par l'algorithme de vérification
– il est difficile, sans la connaissance de la clé secrète du signataire, de produire de nouvelles signatures valides, même en ayant accès aux signatures d'un certain nombre de messages.

Pour étudier la sécurité d'un schéma de signatures, de même que dans le cas du chiffrement, il faut commencer par déterminer précisément les attaques contre lesquelles on veut se prémunir, et quels sont les moyens dont disposent l'attaquant pour les réaliser. Informellement, les objectifs de l'attaquant peuvent être les suivants :

– *cassage total* : il retrouve la clé secrète du signataire,

– *contrefaçon (ou forge) universelle* : il peut signer tous les messages, sans connaître la clé secrète,

– *contrefaçon (ou forge) sélective* : il peut signer un message de son choix,

– *contrefaçon (ou forge) existentielle* : il peut générer un couple message/signature valide.

Pour cela, l'attaquant peut s'engager dans

– une *attaque sans message* : il connaît seulement la clé publique du signataire dont il veut contrefaire la signature,

– une *attaque à messages connus* : il connaît la clé publique du signataire cible, et une liste de couples messages/signatures de ce signataire,

– une *attaque à messages choisis* : il connaît la clé publique du signataire cible, et obtient des signatures de messages de son choix, qu'il peut construire de façon adaptative en fonction de la clé publique et des messages dont il a précédemment obtenu la signature.

D'un point de vue théorique, il suffit d'une fonction à sens unique pour construire un schéma de signatures sûr : c'est ce qu'a montré Rompel dans [210]. Bellare et Micali [19], puis Naor et Yung [179] avaient auparavant montré que l'on pouvait obtenir des signatures grâce à des permutations à trappes et à des permutations à sens unique, respectivement.

Comme pour le chiffrement, ces résultats théoriques n'offrent pas de schémas efficaces. La méthode la plus souvent utilisée pour élaborer un schéma de signatures est d'utiliser le "hacher et signer" (*hash and sign*) : c'est en fait un haché du message qui passe dans la fonction de signature. La taille du message à signer est ainsi diminuée, donc les performances accrues, et cela améliore la sécurité contre les forges existentielles. En général, il est nécessaire, pour prouver la sécurité de telles constructions, de faire l'hypothèse que la fonction de hachage se comporte comme une fonction aléatoire. Ce n'est cependant pas toujours le cas : dans [100], Gennaro, Halevi et Rabin proposent des signatures dont la sécurité repose sur l'hypothèse RSA flexible (voir section 2.1.2), et sur l'existence de fonctions de hachage un peu particulières, "résistantes à la division", mais qu'il est toutefois possible de construire.

1.2.4 Les preuves de sécurité

1.2.4.1 Preuves et réductions

La résistance acharnée d'un schéma cryptographique à l'assaut des cryptanalystes n'est pas un gage de sa sécurité. Au début de l'ère cryptographique, nombre de protocoles étaient présentés avec des preuves heuristiques, ou simplement des arguments de sécurité. Ceux-ci résistaient rarement longtemps aux attaques plus ou moins sophistiquées des cryptanalystes. Les faiblesses de ces schémas provenaient du manque de contexte formel pour l'étude de leur sécurité. De nombreux schémas proposés n'ont été formellement définis que bien après leur publication, et leur sécurité dans ce nouveau modèle contredisait les arguments initiaux.

L'avènement de la *sécurité prouvée*, introduite en 1984 par Goldwasser et Micali [112], a permis avec le temps de remédier à ces approximatifs arguments de sécurité. Dans ce contexte, l'étude de la sécurité d'un cryptosystème est reliée à la théorie de la complexité : ce sont des preuves *réductionnistes* [13] qui exhibent des réductions entre la résolution d'un problème bien identifié et réputé difficile (RSA, logarithme discret,...) et une attaque sur un protocole. Dans un premier temps, de grossières réductions polynomiales étaient proposées, ce qui n'était pas suffisant pour une étude du protocole dans un contexte réel, utilisé en pratique. Dans ces preuves réductionnistes, on contruit un algorithme qui utilise un attaquant contre le système étudié comme une sous-routine pour résoudre un problème difficile. Si la réduction est polynomiale, on conclut qu'attaquer le cryptosystème est au moins aussi dur que de résoudre le problème. Cela sous-entend de définir précisément la signification d'un attaquant en fonction de ses moyens et de ses objectifs. Goldwasser et Micali l'ont formalisée pour le chiffrement dans [112], et Goldwasser, Micali et Rivest pour la signature dans [115, 116]. Cependant, les réductions exhibées, bien que polynomiales, restent souvent impraticables pour résoudre une petite instance. Nous reviendrons sur ces notions dans les parties suivantes sur le chiffrement et la signature.

La *sécurité exacte* [25] ou *sécurité concrète* [185] est un pas supplémentaire vers la possibilité d'analyser finement la sécurité des protocoles utilisés en pratique, et donc vers la *sécurité pratique* comme définie par Pointcheval [196] : on cherche à obtenir une réduction pour laquelle la probabilité de casser le schéma est essentiellement la même que celle d'attaquer le cryptosystème, dans essentiellement le même temps.

1.2.4.2 Les différents modèles

Modèle standard L'étude de la sécurité d'un algorithme de cryptographie asymétrique consiste à montrer que l'existence d'un attaquant mettant en défaut ses propriétés de sécurité en un temps t avec une probabilité ε permet de contruire un algorithme pouvant résoudre des instances quelconques d'un problème réputé difficile en temps $f(t)$ et avec une probabilité $g(t)$, où f et g sont des fonctions (polynomiales). Si $f(t)$ est plus petite que le meilleur algorithme existant pour résoudre le problème sous-jacent, on conclut que le cryptosystème est sûr. Sans hypothèse particulière concernant par exemple les fonctions de hachage utilisées dans le schéma (autre que l'hypothèse minimale de résistance aux collisions), on parle de preuve conduite dans le *modèle standard*. Faire une preuve de sécurité sans hypothèse supplémentaire est souvent difficile, et ne permet pas d'avoir des schémas très efficaces. En particulier, très peu de schémas de signature ou de chiffrement efficaces ont été prouvés sûrs dans le modèle standard. Le schéma de Boneh et Boyen est un élégant contre-exemple [34], bien que (et c'est aussi souvent le cas), l'hypothèse calculatoire sous-jacente ne soit pas particulièrement standard, elle.

Aussi, pour obtenir des preuves pertinentes, il est souvent nécessaire de supposer que certains objets utilisés dans la conception du schéma se comportent "bien" (en essayant de ne pas trop s'éloigner de la réalité) : en particulier les fonctions de hachage (modèle de l'oracle aléatoire) ou la structure du groupe (modèle générique).

Modèle de l'oracle aléatoire Le modèle de l'oracle aléatoire a été introduit par Fiat et Shamir [88] en 1986 puis Bellare et Rogaway [23]. Dans ce modèle, on fait l'hypothèse que les fonctions de hachage utilisées dans le cryptosystème étudié se comportent comme des fonctions aléatoires. Ainsi, dans la preuve, cette fonction est modélisée par un oracle qui produit, à chaque nouvelle requête, une valeur aléatoire. L'oracle renvoie la même valeur à une requête répétée. L'idée est que la connaissance des hachés de certaines valeurs ne puisse pas être, en pratique, exploitée pour prévoir le haché de prochaines valeurs. Dans la preuve, on fait l'hypothèse qu'aucune information ne peut en être déduite. Au début de la preuve, l'entropie de l'oracle est maximale dans l'ensemble des fonctions de $\{0,1\}^n$ dans $\{0,1\}^k$, et à chaque réponse de l'oracle, cette entropie diminue exactement de k bits. L'utilisation de l'oracle aléatoire offre beaucoup de latitude pour faire la preuve de sécurité. Ce modèle, largement répandu et accepté, est parfois contesté. Canetti, Goldreich et Halevi [52, 53] ont montré des contres-exemples liés à une mauvaise mise en place des schémas qu'ils considéraient, et Bellare, Boldyreva et Palacio [15] ont montré des schémas hybrides non instanciables avec des considérations de sécurité ad hoc. Dans tous les cas, ces contre-exemples sont très peu intuitifs et très artificiels.

Modèle générique Le modèle générique est un modèle dans lequel on considère que les opérations arithmétiques n'ont pas de propriétés particulières que l'on pourrait exploiter. La structure algébrique est donc complètement opaque pour un attaquant. Dans ce modèle, lorsqu'un cryptosystème est construit sur un groupe, le groupe sous-jacent est dit *générique*, et l'adversaire a seulement accès à un *encodage* des éléments. Une fonction d'encodage est une application injective σ de $(\mathbb{Z}/q\mathbb{Z},+)$ dans $\{0,1\}^n$, qui encode les éléments du groupe en des suites de bits. Dans un groupe générique, l'encodage est aléatoire. Les premiers algorithmes "génériques" ont été utilisés par Nechaev en 1994 [181] et Shoup en 1997 [227] pour étudier la difficulté du problème du logarithme discret. Ainsi, un algorithme générique $\mathcal{G}$ est un algorithme probabiliste prenant en entrée une liste d'éléments encodés $\mathcal{L} = \{\sigma(x_1),\ldots,\sigma(x_\ell)\}$, et qui, durant son exécution, a accès à un oracle pour obtenir d'autres encodages. $\mathcal{G}$ interroge l'oracle sur un triplet $\{i,j,\pm\}$, où $(i,j) \in [\![1,\ell]\!]$, et celui-ci lui répond $\sigma(x_i \pm x_j)$, que $\mathcal{G}$ rajoute à sa liste $\mathcal{L}$. Des algorithmes comme l'algorithme ρ de Pollard [200] font partie de cette famille, contrairement aux algorithmes du type "calcul d'index".

D'autres modèles, comme le modèle du chiffrement idéal (*ideal cipher model*) existent, mais ils ne nous seront pas utiles par la suite.

Fonctions de hachage Les fonctions de hachage, définies dans le paragraphes 1.1.2 sont un outil fondamental sous-jacent à (presque) tous les cryptosystèmes. La résistance à la seconde préimage est nécessaire pour éviter qu'un signataire malhonnête, ayant authentifié un message m_0 en signant le haché $H(m_0)$, ne puisse faire croire qu'il a signé un message $m_1 \neq m_0$ tel que $H(m_0) = H(m_1)$. Par ailleurs, la résistance aux collisions est une notion tout aussi importante qui est souvent nécessaire pour élaborer une preuve de sécurité dans le modèle standard. Parmi les fonctions de hachage classiques, notons les fonctions MD4 [205] et MD5 [206] de Rivest, ou la famille des SHA [182, 183]. À la *rump session* de Crypto'04, Joux a présenté une collision sur SHA-0 [129], et Wang a présenté des résultats sur MD4, MD5, HAVAL-128 et RIPEMD [242]. Par ailleurs, une attaque sur SHA-1 en 2^{69} opérations a été annoncée[3] par Wang, Yin et Yu [243]. Une grande partie des fonctions de hachage est en train de s'écrouler : bon nombre des cryptosystèmes sont construits sur les fonctions qui sont en train d'être cassées. Comme il y a relativement peu de constructions différentes, l'élaboration de fonctions de hachage rapides et sûres s'avère indispensable.

1.2.5 Preuves interactives à divulgation nulle de connaissance

Si la façon la plus simple de prouver la connaissance d'une donnée en sa possession est de l'exposer, cette solution n'est pas satisfaisante quand la donnée en question est sensible ou secrète. Il est souvent très utile, par exemple pour s'authentifier, de pouvoir prouver que l'on connaît un secret. La question d'une telle preuve, qui ne dévoilerait "rien" du secret, s'est posée très tôt. Une manière classique de s'authentifier consiste en l'utilisation d'un mot de passe. Ce dernier, doit bien entendu rester secret, et le fait est que la façon la plus classique de s'en servir pour s'authentifier est de le révéler ! La question qui intéresse les cryptologues est donc : est-il possible de prouver sa connaissance d'un secret en ne dévoilant aucune information sur celui-ci ? Bien que cette exigence soit paradoxale, en 1985, Goldwasser, Micali et Rackoff [113, 114] démontrent que l'on peut prouver l'appartenance d'un mot à un langage en ne dévoilant aucune autre information sur ce mot. Les *preuves à divulgation nulle de connaissance* ont ensuite été formalisées par Feige, Fiat et Shamir en 1988 [86]. En un mot, les preuves à divulgation nulle de connaissance n'apprennent aucune autre information que la validité de l'assertion. On peut se référer à l'article de Goldreich [106] pour un tour d'horizon des preuves à divulgation nulle de connaissance.

1.2.5.1 Définitions

Dans toute la suite, un *prouveur* $\mathcal{P}$ va vouloir convaincre un *vérificateur* $\mathcal{V}$. Ce sont tous deux des PPTM. Si l'on note x la donnée initiale commune aux deux entités, le résultat d'un protocole interactif entre les deux machines (succès ou échec - 1 ou 0) sera écrit $\langle \mathcal{P}, \mathcal{V} \rangle(x)$.

Nous définissons ici les preuves d'appartenance à un langage, et les preuves de connaissance (voir [44, 201]).

Définition 8 (Preuve d'appartenance à un langage) *Soit L un langage. Un protocole interactif $(\mathcal{P}, \mathcal{V})$ constitue une* preuve d'appartenance à un langage *pour L si les conditions suivantes sont satisfaites :*

- consistance *: un prouveur honnête est accepté avec une probabilité proche de 1 : il existe une fonction négligeable ν telle que*

$$\forall x \in L,\ \Pr\left[\langle \mathcal{P}; \mathcal{V} \rangle(x) = 1\right] > 1 - \nu(|x|),$$

- significativité *: un prouveur malhonnête est accepté avec une probabilité négligeable : il existe une fonction négligeable ν telle que*

$$\forall \tilde{\mathcal{P}}\ \forall x \in \{0,1\}^* \setminus L,\ \Pr\left[\langle \tilde{\mathcal{P}}; \mathcal{V} \rangle(x) = 1\right] < \nu(|x|)$$

On dit alors que le langage L admet un système de preuve interactive pour l'appartenance.

[3] datée du 13 février 2005

Si T est un prédicat à 2 variables, on dit que ω est un *témoin* pour x si $T(\omega, x) = 1$. Le prouveur $\mathcal{P}$ veut convaincre $\mathcal{V}$ de sa connaissance du secret ω. On note $\mathcal{P}(\omega)$ un prouveur connaissant un secret ω.

Définition 9 (Preuve de connaissance) *Un protocole interactif* $(\mathcal{P}, \mathcal{V})$ *constitue une* preuve interactive de connaissance *pour un témoin associé à* x *si les conditions suivantes sont satisfaites :*

- *consistance : un prouveur honnête est accepté avec une probabilité proche de 1 : il existe une fonction négligeable* ν *telle que*

$$\forall x \; \forall \omega \quad T(\omega, x) = 1 \Longrightarrow \Pr\left[\langle \mathcal{P}(\omega); \mathcal{V}\rangle(x) = 1\right] > 1 - \nu(|x|)$$

- *significativité : un prouveur malhonnête (ne connaissant pas le secret) est accepté avec une probabilité négligeable : il existe une fonction négligeable* ν *telle que*

$$\forall \tilde{\mathcal{P}} \; \forall x \in \{0,1\}^* \quad \Pr\left[\langle \tilde{\mathcal{P}}(\star); \mathcal{V}\rangle(x) = 1\right] < \nu(|x|).$$

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

La *vue* du vérificateur, notée $\Xi_{\mathcal{V}}\langle \mathcal{P}(\omega)\rangle(x)$, est la distribution de variables aléatoires décrivant les échanges entre $\mathcal{V}$ et $\mathcal{P}$ lorsque le prouveur opère avec le secret ω associé à x.

Définition 10 (Divulgation nulle de connaissance - zero-knowledge) *Un protocole interactif* $(\mathcal{P}, \mathcal{V})$ *est à divulgation nulle de connaissance si, pour tout vérificateur (éventuellement malhonnête)* $\tilde{\mathcal{V}}$*, il existe un simulateur*[4] *polynomial* $\mathcal{M}_{\tilde{\mathcal{V}}}$ *tel que :*

$$\forall \omega \; \forall x \in \{0,1\}^* \quad \Xi_{\tilde{\mathcal{V}}}\langle \mathcal{P}(\omega)\rangle(x) \sim \mathcal{M}_{\tilde{\mathcal{V}}}(x),$$

où $\sim$ *signifie respectivement parfaitement, statistiquement ou calculatoirement indistinguable. On parle alors respectivement de protocoles parfaitement, statistiquement, ou calculatoirement zero-knowledge.*

Si le simulateur ne dépend pas du vérificateur (c'est un simulateur *universel*), on parle de *black-box* zero-knowledge. C'est une notion plus forte qui s'applique en général aux situations pratiques [117].

On considère souvent la notion plus faible de *divulgation nulle de connaissance face à un vérificateur honnête*, comme c'est le cas dans les exemples qui suivent. Cela signifie qu'il existe un simulateur pour un vérificateur qui suit le protocole, et qui ne s'en écarte pas :

$$\exists \mathcal{M}_{\mathcal{V}} \; : \; \forall \omega \; \forall x \in \{0,1\}^* \quad \Xi_{\mathcal{V}}\langle \mathcal{P}(\omega)\rangle(x) \sim \mathcal{M}_{\mathcal{V}}(x).$$

Théoriquement, il est possible de prouver de façon interactive et de façon *zero-knowledge* l'appartenance d'un mot à un langage dans $\mathcal{NP}$. De même, pour tout langage dans $\mathcal{NP}$, et pour tout mot de ce langage, on peut prouver de façon interactive et de façon *zero-knowledge* la connaissance d'un *témoin* [109].

Goldreich et Krawczyk ont montré en 1990 [108] qu'il est impossible de construire une preuve à divulgation nulle de connaissance en moins de quatre passes : s'il existe une preuve de connaissance à divulgation nulle de connaissance en trois passes pour un problème donné, alors ce problème peut être résolu en temps probabiliste polynomial.

[4] au sens de [113]

1.2.5.2 Exemples

Nous présentons ici deux exemples classiques de preuves qui seront utiles dans les protocoles d'enregistrement de clés ou de confirmation dans les schémas de signatures que nous construirons dans la partie suivante.

Égalité de deux logarithmes discrets La figure 1.1 décrit une preuve d'égalité de deux logarithmes discrets, proposée par Chaum dans [54]. Le groupe $\mathbb{G}$ est d'ordre un grand premier q engendré par P. Le prouveur possède une clé secrète $x \in [\![1, q-1]\!]$ et une clé publique $P_x = xP$. Il va prouver au vérificateur que $\log_P(P_x) = \log_Q(S)$, pour P et Q dans $\mathbb{G}$. Le protocole se déroule donc de la façon suivante : le vérificateur tire au hasard un couple $(a, b) \in [\![1, q-1]\!]^2$ et envoie le point $C = aP + bQ$ au prouveur. Ce dernier tire aléatoirement $r \in [\![1, q-1]\!]$ et construit le point R_1 comme $C + rP$ qui lui sert à masquer son secret dans $R_2 = xR_1$. Le prouveur envoie ces deux points, et le vérificateur découvre a et b. Si l'équation $C = aP + bQ$ est vérifié, le prouveur découvre son aléa r, et le vérificateur accepte la preuve si $R_1 = (a + r)P + bQ$ et $R_2 = (a + r)P_x + S$.

Théorème 1 *Si q est suffisamment grand, la preuve d'égalité de deux logarithmes discrets de Chaum est une preuve d'égalité à divulgation nulle de connaissance face à un vérificateur honnête.*

Prouveur		Vérificateur
		$(a, b) \in_R [\![1, q-1]\!]^2$
	$\xleftarrow{\quad C \quad}$	$C = aP + bQ$
$r \in_R [\![1, q-1]\!]$		
$R_1 = C + rP$		
$R_2 = xR_1$	$\xrightarrow{\quad R_1, R_2 \quad}$	
	$\xleftarrow{\quad a, b \quad}$	
$C \stackrel{?}{=} aP + bQ$	$\xrightarrow{\quad r \quad}$	
		$R_1 \stackrel{?}{=} (a + r)P + bQ$
		$R_2 \stackrel{?}{=} (a + r)P_x + bS$

FIG. 1.1 – Preuve d'égalité de 2 logarithmes discrets : $\log_P(P_x) = \log_Q(S)$

Protocole d'authentification de Schnorr La figure 1.3 décrit le protocole d'authentification de Schnorr [215]. Il fait partie des protocoles d'authentification en 3 passes, formatés selon le schéma de la figure 1.2.

Les entiers p et q sont deux premiers tels que $q \mid p - 1$, et $q > 2^k$, où $k \in \mathbb{N}$ est un paramètre de sécurité. L'élément $g \in (\mathbb{Z}/p\mathbb{Z})^*$ est d'ordre q. Le prouveur possède une clé secrète $x \in \mathbb{Z}/q\mathbb{Z}$, et sa clé publique est $y = g^{-x} \mod p$.

Théorème 2 *Si k est suffisamment grand, le protocole de Schnorr est une preuve de connaissance d'un logarithme discret à divulgation nulle de connaissance face à un vérificateur honnête.*

Des résultats récents plus complets ont été démontrés dans [21] par Bellare et Palacio.

Prouveur		Vérificateur
	$\xrightarrow{\text{engagement}}$	
	$\xleftarrow{\text{défi}}$	
	$\xrightarrow{\text{réponse}}$	

FIG. 1.2 – Protocole d'authentification en 3 passes

Prouveur		Vérificateur
$r \in_R \mathbb{Z}/q\mathbb{Z}$		
$c = g^r \mod p$	$\xrightarrow{c}$	
	$\xleftarrow{e}$	$e \in \mathbb{Z}/2^k\mathbb{Z}$
$d = r - ex \mod q$	$\xrightarrow{d}$	
		$c = g^d y^e \mod p$

FIG. 1.3 – Protocole d'authentification de Schnorr

1.2.5.3 L'heuristique Fiat-Shamir

Quand des échanges entre le prouveur et le vérificateur ne sont pas souhaitables, la question d'une preuve *non interactive* se pose [29, 212]. L'heuristique de Fiat-Shamir [88] concerne la transformation d'une preuve interactive sûre face à un vérificateur honnête en une preuve non interactive sûre dans le modèle de l'oracle aléatoire. Ces preuves sont parfois appelées *signatures de connaissance*. En effet, une signature numérique peut se voir comme une preuve (qui dépend d'un message) de la connaissance de sa clé secrète. En particulier, Pointcheval et Stern ont montré formellement dans [199] qu'un protocole d'authentification en 3 passes peut se transformer en une preuve non interactive dans laquelle le défi est dérivé de l'évaluation d'une fonction de hachage en les paramètres publics et l'engagement.

L'idée est de considérer un vérificateur virtuel dont le défi (la question) dépend du message et de l'engagement, en restant imprédictible. Le défi est donc remplacé par le haché du message et de l'engagement (la donnée de la seule vue du protocole ne peut constituer une preuve non interactive, puisque par définition, elle peut être simulée sans la connaissance du secret).

Les signatures de Schnorr [215, 216], décrites dans la figure 1.4 sont issues de cette transformation. La fonction $h : \{0,1\}^* \longrightarrow \{0,1\}^\ell$ est une fonction de hachage.

En utilisant la technique dite du *forking lemma*, introduit dans [194], Pointcheval et Stern prouvent le théorème suivant :

Théorème 3 ([199]) *Soit $\mathcal{A}$ un attaquant EF-CMA contre le schéma de signature de Schnorr de succès ε en temps τ, après $q_{\mathcal{H}}$ requêtes à l'oracle de hachage et q_Σ à l'oracle de signature. Si $\varepsilon \geq 10(q_\Sigma + 1)(q_\Sigma + q_{\mathcal{H}})2^{-\ell}$, alors le problème du logarithme discret dans un sous-groupe d'ordre premier peut être résolu en temps espéré au plus $120686 q_{\mathcal{H}} \tau \varepsilon^{-1}$.*

1.3 Hypothèses calculatoires

Nous allons ici formaliser et quantifier les notions de difficulté d'un certain nombre de problèmes, et en particulier ceux sur lesquels reposent les nouveaux schémas de signatures présentés

Génération des clés :
- p et q sont deux grands premiers tels que $q \mid p-1$, et $q > 2^k$, où $k \in \mathbb{N}$ est un paramètre de sécurité,
- $g \in (\mathbb{Z}/p\mathbb{Z})^*$ est d'ordre q.
- La clé secrète est $x \in \mathbb{Z}/q\mathbb{Z}$, et la clé publique est $y = g^x \mod p$

Signature : Alice, pour signer m, effectue les opérations suivantes :
- $r \in_R (\mathbb{Z}/q\mathbb{Z})^*$,
- $e = h(y||m)$ et $s = r - ex \mod q$.
- La signature est $\sigma = (e, s)$

Vérification :
- Bob calcule $d = y^e g^s \mod q$ et vérifie que $e = h(d||m)$

FIG. 1.4 – Signature de Schnorr

dans ce mémoire, en commençant par rappeler les problèmes classiques sur lesquels repose la cryptographie à clé publique.

1.3.1 RSA

Le problème RSA est présenté dans l'article fondateur de Rivest, Shamir et Adleman [208]. Il consiste à calculer des racines dans un anneau d'entiers modulo un produit de deux premiers. Nous donnons ici sa définition formelle.

Soit $k \in \mathbb{N}$ un paramètre de sécurité. Dans un premier temps, définissons un *générateur de clés RSA* :

Définition 11 (Générateur de clés RSA) *Soit k un entier. Un* générateur de clés RSA RSAGen *est un algorithme probabiliste efficace prenant en entrée un paramètre de sécurité k et retournant un triplet $(N, e, d) \in \mathbb{N} \times ((\mathbb{Z}/N\mathbb{Z})^*)^2$ tel que le* module N *est le produit de deux premiers distincts p et q satisfaisant $2^{\lfloor (k-1)/2 \rfloor} < p < 2^{\lfloor k/2 \rfloor}$ et $2^{\lfloor (k-1)/2 \rfloor} < q < 2^{\lfloor k/2 \rfloor}$ et tel que l'*exposant de chiffrement e *et l'*exposant de déchiffrement d *vérifient $ed \equiv 1 \mod \varphi(N)$.*

Intuitivement, le problème RSA est le suivant :

Problème RSA [RSA] : soit p et q deux grands entiers premiers distincts de même taille. Étant donné $N = pq$ et un couple $(e, y) \in (\mathbb{Z}/N\mathbb{Z})^* \times \mathbb{Z}/N\mathbb{Z}$ quelconque, calculer $x \in (\mathbb{Z}/N\mathbb{Z})$ tel que $y = x^e \mod N$.

L'hypothèse RSA, qui dit informellement qu'il est difficile, étant donné un module RSA N suffisamment grand, de retrouver la racine e-ième d'un élément de $\mathbb{Z}/N\mathbb{Z}$ est quantifiée dans la définition suivante :

Définition 12 *Soit k un entier, RSAGen un générateur de clés RSA, et $\mathcal{A}$ un attaquant qui prend en entrée un paramètre de sécurité k, un couple (N, e) issu du triplet $(N, e, d) = \mathsf{RSAGen}(k)$, et un élément $y \in \mathbb{Z}/N\mathbb{Z}$. Il retourne un élément $x \in \mathbb{Z}/N\mathbb{Z}$. Considérons l'expérience aléatoire suivante :*

$$
\begin{array}{l}
\boxed{\textit{Expérience } \mathbf{Exp}^{RSA}_{\mathsf{RSAGen},\mathcal{A}}(k)} \\
(N, e, d) \leftarrow \mathsf{RSAGen}(k) \\
\text{params} \leftarrow (N, e) \\
y \overset{R}{\leftarrow} (\mathbb{Z}/N\mathbb{Z})^* \\
x \leftarrow \mathcal{A}(k, \text{params}, y) \\
\textit{Retourner } 1 \textit{ si } x^e \equiv y \mod N \textit{ et } 0 \textit{ sinon}
\end{array}
$$

On définit le succès *de $\mathcal{A}$ pour résoudre le problème RSA comme étant*

$$\mathbf{Succ}^{RSA}_{\mathsf{RSAGen},\mathcal{A}}(k) = \Pr\left[\mathbf{Exp}^{RSA}_{\mathsf{RSAGen},\mathcal{A}}(k) = 1\right].$$

Étant donné $\tau \in \mathbb{N}$ et $\varepsilon \in [0,1]$, le problème RSA est dit (k,τ,ε)-difficile s'il n'existe pas d'adversaire $\mathcal{D}$ fonctionnant en temps τ et ayant un succès $\mathbf{Succ}^{RSA}_{\mathsf{RSAGen},\mathcal{A}}(k) \geq \varepsilon$.

Définition 13 (Hypothèse RSA) *Pour tous entiers positifs c et c', et pour k suffisamment grand, le problème RSA est $(k, k^c, k^{-c'})$-difficile.*

Plus généralement, quand on considérera un problème $\mathbb{P}$, supposé difficile, et que l'on aura défini et quantifié le succès ou l'avantage d'un attaquant contre celui-ci de façon analogue à la définition 12, on fera l'hypothèse que pour tous entiers positifs c et c', et pour k suffisamment grand, il est $(k, k^c, k^{-c'})$-difficile.

1.3.2 Logarithme discret

L'autre problème classique de la cryptographie à clé publique et cité dès le papier de Diffie et Hellman en 1976 [76] est celui du logarithme discret. Nous utiliserons par la suite une notation additive pour suggérer comme groupe sous-jacent le groupe des points d'une courbe elliptique (ou la jacobienne d'une courbe algébrique), dont la loi de groupe est traditionnellement notée additivement. La plupart du temps, on choisit un groupe d'ordre premier pour mettre en place un cryptosystème basé sur le logarithme discret. En effet, si le groupe est d'ordre n quelconque, on peut ramener le problème du logarithme discret dans ce groupe à des problèmes de logarithmes discrets dans les sous-groupes d'ordre premier divisant n (si on connait la factorisation de n). La méthode, basée sur le théorème des restes chinois, a été décrite dans l'article de Pohlig et Hellman [193]. Ainsi, on définit un générateur de groupe d'ordre premier.

Définition 14 (Générateur de groupe d'ordre premier) *Soit k un entier. Un* générateur de groupe d'ordre premier *DLGen est un algorithme efficace prenant en entrée un paramètre de sécurité k et retournant un triplet $(q, P, \mathbb{G})$ tel que q est un nombre premier satisfaisant $2^{k-1} < q < 2^k$ et $\mathbb{G}$ est un groupe d'ordre q engendré par l'élément P.*

Intuitivement, le problème du logarithme discret est le suivant :

Problème du logarithme discret [DL] : soit $\mathbb{G}$ un groupe ayant un grand ordre premier q, et $P \in \mathbb{G}$ un générateur de $\mathbb{G}$. Étant donné $Q \in \mathbb{G}$, calculer $x \in [\![1, q-1]\!]$ tel que $Q = xP$. L'entier x est le logarithme discret de Q en base P et est noté $\log_P(Q)$.

Des exemples de tels groupes cryptographiquement intéressants sont le groupe multiplicatif d'un corps fini, le groupe des points d'une courbe elliptique, un ordre dans un corps de nombres,... Ce ne peut pas être, par exemple, le groupe $\mathbb{Z}/p\mathbb{Z}$, où p est un entier quelconque, muni de la loi d'addition : dans ce groupe le logarithme discret peut être calculé trivialement par l'algorithme d'Euclide. La difficulté du logarithme discret est différente suivant le groupe : il est réputé particulièrement difficile dans le groupe des points d'une courbe elliptique ; pour la plupart des autres groupes, il existe un algorithme sous-exponentiel pour le résoudre.

Définition 15 (DL) *Soit k un entier, DLGen un générateur de groupe d'ordre premier et $\mathcal{A}$ un attaquant prenant en entrée un paramètre de sécurité k, un triplet $(q, P, \mathbb{G}) = \mathsf{DLGen}(k)$ et un élément $Q \in \mathbb{G}$. Considérons l'expérience aléatoire suivante :*

$\boxed{\textit{Expérience } \mathbf{Exp}^{DL}_{\mathsf{DLGen},\mathcal{A}}(k)}$
$(q, P, \mathbb{G}) \leftarrow \mathsf{DLGen}(k)$
$\text{params} \leftarrow (q, P, \mathbb{G})$
$x \xleftarrow{R} [\![1, q-1]\!]$
$Q \leftarrow xP$
$x' \leftarrow \mathcal{A}(k, \text{params}, Q)$
Retourner 1 *si* $x' = x$ *et* 0 *sinon.*

On définit le succès *de $\mathcal{A}$ pour résoudre le problème du logarithme discret comme étant*

$$\mathbf{Succ}^{DL}_{\mathsf{DLGen},\mathcal{A}}(k) = \Pr\left[\mathbf{Exp}^{DL}_{\mathsf{DLGen},\mathcal{A}}(k) = 1\right].$$

Étant donné $\tau \in \mathbb{N}$ et $\varepsilon \in [0, 1]$, le problème du logarithme discret est dit (k, τ, ε)-difficile s'il n'existe pas d'adversaire $\mathcal{A}$ qui, en temps τ, a un succès $\mathbf{Succ}^{DL}_{\mathsf{DLGen},\mathcal{A}}(k) \geq \varepsilon$.

1.3.3 Diffie-Hellman et ses variantes

1.3.3.1 Introduction

Les hypothèses Diffie-Hellman sont liées au protocole d'échange de clé publié dans l'article fondateur de la cryptographie asymétrique [76], décrit dans la figure 1.5 (il est à noter que cet échange de clé n'assure aucune authenticité).

Alice		Bob
$a \in [\![1, q-1]\!]$ $P_A = aP$		$b \in [\![1, q-1]\!]$ $P_B = bP$
	$\xrightarrow{\quad P_A \quad}$	
	$\xleftarrow{\quad P_B \quad}$	
$K = aP_B = abP$		$K = bP_A = abP$

FIG. 1.5 – Échange de clé Diffie-Hellman

Un attaquant passif (une oreille indiscrète, Ève), pour casser totalement le protocole d'échange de clé doit, à partir de aP et bP, calculer le point abP. L'hypothèse Diffie-Hellman calculatoire (CDH) dit qu'il n'existe justement pas d'algorithme efficace pour faire ce calcul. Cette hypothèse est plus forte que l'hypothèse du logarithme discret, puisque ce problème Diffie-Hellman calculatoire est plus facile que celui du logarithme discret. En effet, si l'on sait extraire a de aP, il est facile, étant donné aP et bP d'extraire a et de calculer $a(bP) = abP$. Cependant, pour des besoins cryptographiques, il est souhaitable qu'Ève n'apprenne *aucune* information sur la clé K. Par exemple, si l'on se place dans un sous-groupe d'ordre q de $(\mathbb{Z}/p\mathbb{Z})^*$ engendré par g, où p et q sont des premiers, il est facile, étant donnés g^a et g^b, de calculer le symbole de Jacobi de g^{ab} et donc de connaître le bit de poids faible de g^{ab}. Ainsi, si l'on veut utiliser certains bits de la clé Diffie-Hellman, la difficulté du problème calculatoire Diffie-Hellman n'assure pas qu'Ève ne puisse pas prédire certains de ces bits. Sous l'hypothèse Diffie-Hellman calculatoire seule, Alice et Bob peuvent calculer un bit hardcore grâce à abP (si Ève pouvait le calculer, elle pourrait calculer tout abP). Ainsi, si l'on veut partager un secret de k bits grâce à ce protocole, il faudrait le répéter k fois, sous la seule hypothèse CDH. Par contre, sous l'hypothèse Diffie-Hellman décisionnelle (DDH), qui consiste à supposer qu'il est difficile de vérifier qu'une solution du problème Diffie-Hellman calculatoire en est bien une, on peut prouver qu'il est possible, en une seule passe de ce protocole, d'extraire $n/2$ bits qu'Ève ne pourra distinguer d'une suite aléatoire. Ceci se fait en hachant la clé de n bits en une empreinte de $n/2$ bits.

Il s'avère que les algorithmes *génériques* pour résoudre les problèmes Diffie-Hellman calculatoire et décisionnel sont des algorithmes génériques de calculs de logarithme discret : Nechaev en 1994 [181] et Shoup en 1997 [227] ont montré des bornes inférieures pour calculer un logarithme discret. Ils montrent qu'il n'existe pas d'algorithme *générique* efficace pour DDH qui fonctionne dans *tous* les groupes. Maurer et Wolf en 1998 [160] ont montré qu'il n'existe pas d'algorithme générique efficace qui réduise CDH à DDH. L'article de Maurer et Wolf [162] résume un certain nombre de résultats de sécurité concernant le protocole Diffie-Hellman. Leur article de 1999 [161] fait le lien entre CDH et le logarithme discret, ainsi que celui de den Boer [71] : dans certains groupes et sous certaines conditions, il existe une réduction polynomiale de CDH vers DL.

Dans la suite, nous donnons les définitions formelles qui quantifient les hypothèses Diffie-Hellman calculatoire et décisionnelle.

1.3.3.2 Diffie-Hellman calculatoire

Le problème Diffie-Hellman calculatoire, qui peut se voir comme une version à trappe du logarithme discret est donc le suivant :

Problème Diffie-Hellman calculatoire [CDH] : soit $\mathbb{G}$ un groupe ayant un grand ordre premier q, et $P \in \mathbb{G}$ un générateur de $\mathbb{G}$. Étant donné $(X, Y) \in \mathbb{G}^2$, calculer l'élément $Z = \log_P(X)\log_P(Y)P$.

Définition 16 (CDH) *Soit k un entier, DLGen un générateur de groupe d'ordre premier et $\mathcal{A}$ un attaquant prenant en entrée un paramètre de sécurité k, un triplet $(q, P, \mathbb{G}) = \mathsf{DLGen}(k)$, et un couple $(X, Y) \in \mathbb{G}^2$ et renvoyant un élément $Z \in \mathbb{G}$. Considérons l'expérience aléatoire suivante :*

$$\boxed{\textit{Experiment } \mathbf{Exp}^{\mathsf{cdh}}_{\mathsf{DLGen},\mathcal{A}}(k)}$$
$$\begin{array}{l}
(q, P, \mathbb{G}) \leftarrow \mathsf{DLGen}(k) \\
\text{params} \leftarrow (q, P, \mathbb{G}) \\
x \xleftarrow{R} [\![1, q-1]\!],\ X \leftarrow xP \\
y \xleftarrow{R} [\![1, q-1]\!],\ Y \leftarrow yP \\
Z \leftarrow \mathcal{A}(k, \text{params}, X, Y) \\
\textit{Retourner } 1 \textit{ si } Z = xyP \textit{ et } 0 \textit{ sinon.}
\end{array}$$

On définit le succès de $\mathcal{A}$ pour résoudre la version calculatoire du problème Diffie-Hellman comme étant

$$\mathbf{Succ}^{\mathsf{cdh}}_{\mathsf{DLGen},\mathcal{A}}(k) = \Pr\left[\mathbf{Exp}^{\mathsf{cdh}}_{\mathsf{DLGen},\mathcal{A}}(k) = 1\right].$$

Étant donné $\tau \in \mathbb{N}$ et $\varepsilon \in [0,1]$, le problème CDH est dit (k,τ,ε)-difficile s'il n'existe pas d'attaquant $\mathcal{A}$ qui, en temps τ, a un succès $\mathbf{Succ}^{\mathsf{cdh}}_{\mathit{DLGen},\mathcal{A}}(k) \geq \varepsilon$.

1.3.3.3 Diffie-Hellman décisionnel

Sommairement, le problème Diffie-Hellman décisionnel est le suivant :

Problème Diffie-Hellman décisionnel [DDH] : soit $\mathbb{G}$ un groupe ayant un grand ordre premier q, et $P \in \mathbb{G}$ un générateur de $\mathbb{G}$. Étant donné $(X, Y, Z) \in \mathbb{G}^3$, décider si $Z = \log_P(X)\log_P(Y)P$.

Informellement, sous l'hypothèse Diffie-Hellman décisionnelle, il n'existe pas d'algorithme efficace qui distingue les deux distributions $\{aP, bP, abP\}$ et $\{aP, bP, cP\}$ pour a, b, c pris au hasard dans $[\![1, q-1]\!]$.

Définition 17 (DDH) *Soit k un entier, DLGen un générateur de groupe d'ordre premier et $\mathcal{A}$ un attaquant prenant en entrée un paramètre de sécurité k, un triplet $(q, P, \mathbb{G}) = \mathit{DLGen}$, un triplet $(X, Y, Q) \in \mathbb{G}^3$ et renvoyant un bit $b \in \{0,1\}$. Considérons les expériences aléatoires suivantes, où $r \in \{0,1\}$:*

$$\boxed{\textit{Experiment } \mathbf{Exp}^{\mathsf{ddh-r}}_{\mathit{DLGen},\mathcal{A}}(k)}$$
$$\begin{array}{l}
(q, P, \mathbb{G}) \leftarrow \mathit{DLGen}(k) \\
\text{params} \leftarrow (q, P, \mathbb{G}) \\
x \xleftarrow{R} [\![1, q-1]\!],\ X \leftarrow xP \\
y \xleftarrow{R} [\![1, q-1]\!],\ Y \leftarrow yP \\
\textit{Si } r = 0 \textit{ faire } Q \leftarrow xyP \\
\textit{Si } r = 1 \textit{ faire } Q \xleftarrow{R} \mathbb{G} \\
b \leftarrow \mathcal{A}(k, \text{params}, X, Y, Q) \\
\textit{Retourner } b.
\end{array}$$

On définit l'avantage de $\mathcal{A}$ pour résoudre la version décisionnelle du problème Diffie-Hellman comme étant

$$\mathbf{Adv}^{\mathsf{ddh}}_{\mathit{DLGen},\mathcal{A}}(k) = \left| Pr\left[\mathbf{Exp}^{\mathsf{ddh-1}}_{\mathit{DLGen},\mathcal{A}}(k) = 1\right] - Pr\left[\mathbf{Exp}^{\mathsf{ddh-0}}_{\mathit{DLGen},\mathcal{A}}(k) = 1\right]\right|.$$

Étant donné $\tau \in \mathbb{N}$ et $\varepsilon \in [0,1]$, le problème DDH est dit (k,τ,ε)-difficile s'il n'existe pas d'attaquant $\mathcal{A}$ qui, en temps τ, a un avantage $\mathbf{Adv}^{\mathsf{ddh}}_{\mathit{DLGen},\mathcal{A}}(k) \geq \varepsilon$.

On peut se référer à l'article de Boneh [32], centré sur le problème DDH, pour en savoir plus.

1.3.3.4 Gap Diffie-Hellman

L'introduction des problèmes "*gap*" par Okamoto et Pointcheval a été motivée par la sécurité de REACT [189] en 2001. La même année, Okamoto et Pointcheval ont formalisé ces problèmes dans [188] et ont prouvé la sécurité du schéma de signatures indéniables de Chaum et van Antwerpen [57] (et sa version corrigée [54]). La question de la sécurité de ce schéma était ouverte depuis 10 ans[5]. Intuitivement, un problème *gap* consiste à résoudre un problème calculatoire

[5]Cependant, Ogata, Kurosawa et Heng ont récemment amélioré cette preuve dans [184]. Ils montré que la sécurité des signatures de Chaum et van Antwerpen repose en fait sur CDH

à l'aide d'un oracle pour le problème décisionnel associé (ou un problème décisionnel proche). Ainsi, le problème Gap Diffie-Hellman peut se formuler de la façon suivante :

Problème Gap Diffie-Hellman [GDH] : soit $\mathbb{G}$ un groupe ayant un grand ordre premier q, et $P \in \mathbb{G}$ un générateur de $\mathbb{G}$. Étant donné $(X,Y) \in \mathbb{G}^2$, calculer l'élément $Z = \log_P(X)\log_P(Y)P$ à l'aide d'un oracle pour le problème décisionnel Diffie-Hellman, noté $\mathcal{O}_{DDH}$.

Lorsqu'il est interrogé sur un quadruplet (P, aP, bP, cP), l'oracle $\mathcal{O}_{DDH}$ indique si ce quadruplet est un quadruplet Diffie-Hellman valide, *i.e.* si $c = ab$.

Définition 18 (GDH) *Soit k un entier, DLGen un générateur de groupe d'ordre premier et $\mathcal{A}$ un attaquant prenant en entrée un paramètre de sécurité k, un triplet $(q, P, \mathbb{G}) = \mathsf{DLGen}(k)$, et un couple $(X,Y) \in \mathbb{G}^2$. Il retourne un élément $Z \in \mathbb{G}$, en ayant accès à un oracle décisionnel. Considérons l'expérience aléatoire suivante :*

$$\boxed{\textit{Experiment } \mathbf{Exp}^{\mathsf{gdh}}_{\mathsf{DLGen},\mathcal{A}}(k)}$$
$$\begin{array}{l}
(q, P, \mathbb{G}) \leftarrow \mathsf{DLGen}(k) \\
\text{params} \leftarrow (q, P, \mathbb{G}) \\
x \stackrel{R}{\leftarrow} [\![1, q-1]\!],\ X \leftarrow xP \\
y \stackrel{R}{\leftarrow} [\![1, q-1]\!],\ Y \leftarrow yP \\
Z \leftarrow \mathcal{A}^{\mathcal{O}_{DDH}}(k, \text{params}, X, Y) \\
\textit{Retourner } 1 \textit{ si } Z = xyP \textit{ et } 0 \textit{ sinon}
\end{array}$$

On définit le succès de $\mathcal{A}$ pour résoudre le problème Gap Diffie-Hellman comme étant

$$\mathbf{Succ}^{\mathsf{gdh}}_{\mathsf{DLGen},\mathcal{A}}(k) = \Pr\left[\mathbf{Exp}^{\mathsf{gdh}}_{\mathsf{DLGen},\mathcal{A}}(k) = 1\right]$$

Étant donné $\tau \in \mathbb{N}$ et $\varepsilon \in [0,1]$, le problème GDH est dit (k, τ, ε)-difficile s'il n'existe pas d'attaquant $\mathcal{A}$ qui, en temps τ, a un succès $\mathbf{Succ}^{\mathsf{gdh}}_{\mathsf{DLGen},\mathcal{A}}(k) \geq \varepsilon$.

Bien sûr, les problèmes *gap* se généralisent à tout autre type de problème. On a parfois besoin de variantes où l'oracle décisionnel est moins puissant que l'oracle pour le problème décisionnel associé au problème calculatoire que l'on souhaite résoudre. Dans le cas des problèmes Diffie-Hellman, un oracle décisionnel pour un problème *statique*[6] [46] peut souvent suffire.

1.3.4 Diffie-Hellman et applications bilinéaires

L'apparition des applications bilinéaires en cryptologie date de l'échange de clé tripartite de Joux [127], décrit dans la figure 1.6.

Apportant un degré de liberté supplémentaire pour la création de nouveaux cryptosystèmes, elles ont aussi apporté une multitude de nouveaux problèmes algorithmiques. Nous présentons tout d'abord les problèmes les plus naturels, puis nous donnons des problèmes moins standards.

Nous rappelons tout d'abord la définition d'une application bilinéaire admissible, qui sera une primitive de base pour les schémas de signatures que nous présenterons par la suite.

[6]le problème statique Diffie-Hellman consiste, étant donné P, xP dans $\mathbb{G}$ fixés, et un élément aléatoire yP de $\mathbb{G}$, à calculer xyP.

Définition 19 (Application bilinéaire admissible) *Soit* $(\mathbb{G}_0, +)$ *et* $(\mathbb{G}_1, +)$ *deux groupes d'un même ordre premier* q*, engendrés respectivement par* P_0 *et* P_1*, et* $(\mathbb{H}, \times)$ *un troisième groupe d'ordre* q*. L'application* $e : \mathbb{G}_0 \times \mathbb{G}_1 \longrightarrow \mathbb{H}$ *est dite* application bilinéaire admissible *si elle vérifie les propriétés suivantes :*
- *bilinéarité, i.e.* $e(aP, bQ) = e(P, Q)^{ab}$ *pour tout* $(a, b) \in \mathbb{Z}^2$ *et* $(P, Q) \in \mathbb{G}_0 \times \mathbb{G}_1$*,*
- *non dégénérescence, i.e.* $e(P_0, P_1) \neq 1$*,*
- *calculabilité, i.e. il existe un algorithme efficace pour calculer* $e(P, Q)$ *pour tout couple* $(P, Q) \in \mathbb{G}_0 \times \mathbb{G}_1$*.*

***Remarque* 1** *Pour la plupart des applications cryptographiques, il est important de connaître et de pouvoir calculer l'isomorphisme* $\mathbb{G}_0 \simeq \mathbb{G}_1$*. On peut donc en général supposer* $\mathbb{G}_0 = \mathbb{G}_1 = \mathbb{G}$*, ce que l'on fera à chaque fois que ce sera possible.*

Alice	Bob	Cindy
$a \in [\![1, q-1]\!]$ $P_A = aP$ $K = e(P_B, P_C)^a = e(P, P)^{abc}$	$b \in [\![1, q-1]\!]$ $P_B = bP$ $K = e(P_A, P_C)^b = e(P, P)^{abc}$	$c \in [\![1, q-1]\!]$ $P_C = cP$ $K = e(P_B, P_A)^c = e(P, P)^{abc}$

FIG. 1.6 – Échange de clé Diffie-Hellman tripartite

L'existence de groupes munis de cette application bilinéaire a une conséquence primordiale sur les problèmes Diffie-Hellman sous-jacents. Dans ces groupes, Joux et Nguyen ont montré dans [131] que l'on peut décider si un quadruplet (P, aP, bP, cP) est un quadruplet Diffie-Hellman valide. En effet
$$c = ab \Longleftrightarrow e(aP, bP) = e(cP, P).$$
Ils exhibent ainsi des groupes dans lesquels le problème calculatoire est difficile, alors que le problème décisionnel est facile. Il est donc nécessaire de préciser les problèmes de type Diffie-Hellman induits par la présence de ces applications bilinéaires.

1.3.4.1 Diffie-Hellman bilinéaire

Dans cette section, nous nous intéressons donc au problème Diffie Hellman bilinéaire (BDH[7]). Nous détaillons les hypothèses Diffie-Hellman bilinéaires naturelles, en commençant par définir un générateur de groupe BDH. Soit $k \in \mathbb{N}$ un paramètre de sécurité.

Définition 20 (Générateur de groupe BDH [36]) *Un* générateur de groupe BDH BDHGen *est un algorithme efficace prenant en entrée un paramètre de sécurité* k *et renvoyant un 7-uplet* $(q, P_0, \mathbb{G}_0, P_1, \mathbb{G}_1, \mathbb{H}, e)$ *tel que* q *est un premier vérifiant* $2^{k-1} < q < 2^k$*, les groupes* $\mathbb{G}_0, \mathbb{G}_1$ *et* $\mathbb{H}$ *sont d'ordre* q*,* P_0 *et* P_1 *engendrent respectivement* $\mathbb{G}_0$ *et* $\mathbb{G}_1$ *et l'application* $e : \mathbb{G}_0 \times \mathbb{G}_1 \longrightarrow \mathbb{H}$ *est une application bilinéaire admissible.*

Le générateur de groupe BDH BDHGen *est dit* symétrique *si, pour tout paramètre de sécurité* k*, il renvoit un 7-uplet* $(q, P_0, \mathbb{G}_0, P_1, \mathbb{G}_1, \mathbb{H}, e)$ *tel que* $\mathbb{G}_0 = \mathbb{G}_1$ *et* $P_0 = P_1$*.*

[7] à ne pas confondre avec B.D.-H. = *Before Diffie-Hellman*

Informellement, le problème bilinéaire Diffie-Hellman est le suivant :

Problème bilinéaire Diffie-Hellman [CBDH] : soit $\mathbb{G}_0$ et $\mathbb{G}_1$ deux groupes ayant un grand ordre premier q, engendrés respectivement par P_0 et P_1. Étant donné $(X, Y, Z) \in \mathbb{G}_0^2 \times \mathbb{G}_1$, calculer $e(P_0, P_1)^{\log_{P_0}(X)\log_{P_1}(Y)\log_{P_1}(Z)}$.

Nous commençons par donner la définition de la version décisionnelle :

Problème bilinéaire Diffie-Hellman décisionnel [DBDH] : soit $\mathbb{G}_0$ et $\mathbb{G}_1$ deux groupes ayant un grand ordre premier q, engendrés respectivement par P_0 et P_1. Étant donné $(X, Y, Z, h) \in \mathbb{G}_0^2 \times \mathbb{G}_1 \times \mathbb{H}$, décider si $h = e(P_0, P_1)^{\log_{P_0}(X)\log_{P_1}(Y)\log_{P_1}(Z)}$.

Définition 21 (DBDH) *Soit k un entier, BDHGen un générateur de groupe BDH et $\mathcal{A}$ un attaquant prenant en entrée un paramètre de sécurité k, un 7-uplet $(q, P_0, \mathbb{G}_0, P_1, \mathbb{G}_1, \mathbb{H}, e) =$ BDHGen(k) et un quadruplet $(X, Y, Z, h) \in \mathbb{G}_0^2 \times \mathbb{G}_1 \times \mathbb{H}$, et renvoyant un bit $b \in \{0, 1\}$. Considérons les expériences aléatoires suivantes, où $r \in \{0, 1\}$:*

$$\boxed{\textit{Experiment } \mathbf{Exp}^{\mathsf{dbdh-r}}_{\mathit{BDHGen},\mathcal{A}}(k)}$$

$$\begin{array}{l}
(q, P_0, \mathbb{G}_0, P_1, \mathbb{G}_1, \mathbb{H}, e) \leftarrow \mathit{BDHGen}(k) \\
\text{params} \leftarrow (q, P_0, \mathbb{G}_0, P_1, \mathbb{G}_1, \mathbb{H}, e) \\
x \xleftarrow{R} [\![1, q-1]\!],\ X \leftarrow xP_0 \\
y \xleftarrow{R} [\![1, q-1]\!],\ Y \leftarrow yP_0 \\
z \xleftarrow{R} [\![1, q-1]\!],\ Z \leftarrow zP_1 \\
\textit{Si } r = 0 \textit{ faire } h \leftarrow e(P_0, P_1)^{xyz} \\
\textit{Si } r = 1 \textit{ faire } h \xleftarrow{R} \mathbb{H} \\
b \leftarrow \mathcal{A}(k, \text{params}, X, Y, Z, h) \\
\textit{Retourner } b.
\end{array}$$

On définit l'avantage de $\mathcal{A}$ pour résoudre la version décisionnelle du problème bilinéaire Diffie-Hellman comme étant

$$\mathbf{Adv}^{\mathsf{ddh}}_{\mathit{BDHGen},\mathcal{A}}(k) = \left| Pr\left[\mathbf{Exp}^{\mathsf{dbdh-1}}_{\mathit{BDHGen},\mathcal{A}}(k) = 1\right] - Pr\left[\mathbf{Exp}^{\mathsf{dbdh-0}}_{\mathit{BDHGen},\mathcal{A}}(k) = 1\right]\right|.$$

Étant donné $\tau \in \mathbb{N}$ et $\varepsilon \in [0, 1]$, le problème DBDH est dit (k, τ, ε)-difficile s'il n'existe pas d'attaquant $\mathcal{A}$ qui, en temps τ, a un avantage $\mathbf{Adv}^{\mathsf{dbdh}}_{\mathit{BDHGen},\mathcal{A}}(k) \geq \varepsilon$.

Nous pouvons ainsi donner les définitions de la version calculatoire (CBDH) et de la version *gap* (GBDH).

Définition 22 (CBDH,GBDH) *Soit k un entier, BDHGen un générateur de groupe BDH et $\mathcal{A}$ un attaquant qui prend en entrée un paramètre de sécurité k, un 7-uplet $(q, P_0, \mathbb{G}_0, P_1, \mathbb{G}_1, \mathbb{H}, e)$ généré par BDHGen, un triplet $(X, Y, Z) \in \mathbb{G}_0^2 \times \mathbb{G}_1$ et renvoie un élément $h \in \mathbb{H}$. Considérons les expériences aléatoires suivantes :*

$\boxed{\textit{Experiment } \mathbf{Exp}^{\mathsf{cbdh}}_{\mathit{BDHGen},\mathcal{A}}(k)}$	$\boxed{\textit{Experiment } \mathbf{Exp}^{\mathsf{gbdh}}_{\mathit{BDHGen},\mathcal{A}}(k)}$
$(q, P_0, \mathbb{G}_0, P_1, \mathbb{G}_1, \mathbb{H}, e) \leftarrow \mathit{BDHGen}(k)$	$(q, P_0, \mathbb{G}_0, P_1, \mathbb{G}_1, \mathbb{H}, e) \leftarrow \mathit{BDHGen}(k)$
$\text{params} \leftarrow (q, P_0, \mathbb{G}_0, P_1, \mathbb{G}_1, \mathbb{H}, e)$	$\text{params} \leftarrow (q, P_0, \mathbb{G}_0, P_1, \mathbb{G}_1, \mathbb{H}, e)$
$x \xleftarrow{R} [\![1, q-1]\!],\ X \leftarrow xP_0$	$x \xleftarrow{R} [\![1, q-1]\!],\ X \leftarrow xP_0$
$y \xleftarrow{R} [\![1, q-1]\!],\ Y \leftarrow yP_0$	$y \xleftarrow{R} [\![1, q-1]\!],\ Y \leftarrow yP_0$
$z \xleftarrow{R} [\![1, q-1]\!],\ Z \leftarrow zP_1$	$z \xleftarrow{R} [\![1, q-1]\!],\ Z \leftarrow zP_1$
$h \leftarrow \mathcal{A}(k, \text{params}, X, Y, Z)$	$h \leftarrow \mathcal{A}^{\mathcal{O}_{DBDH}}(k, \text{params}, X, Y, Z)$
Retourner 1 *si* $h = e(P_0, P_1)^{xyz}$,	*Retourner* 1 *si* $h = e(P_0, P_1)^{xyz}$,
0 *sinon*	0 *sinon*

*On définit alors le succès de $\mathcal{A}$ pour résoudre le problème CBDH (*resp. *GBDH) comme étant*

$$\mathbf{Succ}^{\mathsf{cbdh}}_{\mathit{BDHGen},\mathcal{A}}(k) = \Pr\left[\mathbf{Exp}^{\mathsf{cbdh}}_{\mathit{BDHGen},\mathcal{A}}(k) = 1\right] \quad \Big| \quad \mathbf{Succ}^{\mathsf{gbdh}}_{\mathit{BDHGen},\mathcal{A}}(k) = \Pr\left[\mathbf{Exp}^{\mathsf{gbdh}}_{\mathit{BDHGen},\mathcal{A}}(k) = 1\right]$$

Étant donné $\tau \in \mathbb{N}$ et $\varepsilon \in [0,1]$, le problème CBDH (respectivement GBDH) est dit (k, τ, ε)-difficile s'il n'existe pas d'attaquant $\mathcal{A}$ qui, en temps τ, a un succès $\mathbf{Succ}^{\mathsf{cbdh}}_{\mathit{BDHGen},\mathcal{A}}(k) \geq \varepsilon$ (respectivement $\mathbf{Succ}^{\mathsf{gbdh}}_{\mathit{BDHGen},\mathcal{A}}(k) \geq \varepsilon$).

Il existe de légères variantes du problème Diffie-Hellman bilinéaire quand les groupes $\mathbb{G}_0$ et $\mathbb{G}_1$ sont différents, et que l'on n'a pas explicitement l'isomorphisme entre les deux groupes. En particulier dans les groupes BDH, on peut considérer des variantes de CDH et DDH, induites par la présence de l'application bilinéaire : par exemple, étant donné P_1 et aP_1 dans $\mathbb{G}_1$ et X dans $\mathbb{G}_0$, calculer $a_1X \in \mathbb{G}_0$. Ce problème est appelé co-CDH [38] ; il est équivalent à CDH si $\mathbb{G}_0 = \mathbb{G}_1$.

1.3.4.2 Des variantes exotiques

La sécurité d'un grand nombre de cryptosystèmes ayant pour brique de base une application bilinéaire admissible, ne se réduit pas (hélas) aux problèmes précédemment décrits. Il est souvent nécessaire de faire appel à des hypothèses beaucoup plus fortes, moins intuitives. Nous décrivons maintenant un certain nombre de ces hypothèses, qui sont liées aux schémas de signatures que nous proposons par la suite.

ℓ-SDH L'hypothèse *ℓ-strong Diffie-Hellman* (ℓ-SDH) a été introduite par Boneh et Boyen dans [34] pour prouver la sécurité dans le modèle standard de leur schéma de signatures courtes. Cette hypothèse, non standard, est considérée comme l'analogue pour Diffie-Hellman de l'hypothèse RSA flexible. Boneh et Boyen donnent une intuition de la difficulté du problème ℓ-SDH en calculant une borne inférieure de la complexité de sa résolution dans le modèle générique. Le problème ℓ-SDH est le suivant :

Problème ℓ-Diffie-Hellman flexible (ℓ-SDH) : soit $x \in [\![1, q-1]\!]$. Étant donné un entier ℓ et $(P, xP, x^2P, \ldots, x^\ell P) \in \mathbb{G}^{\ell+1}$, calculer un couple $\big((x+h)^{-1}P, h\big)$ dans $\mathbb{G} \times [\![1, q-1]\!]$ pour un certain $h \in [\![1, q-1]\!]$.

L'hypothèse Diffie-Hellman flexible est définie de la façon suivante :

Définition 23 (ℓ-SDH) *Soit k un entier, BDHGen un générateur de groupe BDH et $\mathcal{A}$ un attaquant qui prend en entrée un paramètre de sécurité k, un 7-uplet $(q, P_0, \mathbb{G}_0, P_1, \mathbb{G}_1, \mathbb{H}, e) =$*

$\mathsf{BDHGen}(k)$ et un $(\ell+2)$-uplet $(P_0, P_1, Q_1, \ldots, Q_\ell) \in \mathbb{G}_0 \times \mathbb{G}_1^{\ell+1}$ et qui renvoie un couple $(c, X) \in \mathbb{Z}/q\mathbb{Z}^* \times \mathbb{G}_0$. Considérons les expériences aléatoires suivantes :

$$\boxed{\textit{Experiment } \mathbf{Exp}^{\ell-\mathsf{sdh}}_{\mathsf{BDHGen},\mathcal{A}}(k)}$$
$(q, P_0, \mathbb{G}_0, P_1, \mathbb{G}_1, \mathbb{H}, e) \leftarrow \mathsf{BDHGen}(k)$
$\text{params} \leftarrow (q, P_0, \mathbb{G}_0, P_1, \mathbb{G}_1, \mathbb{H}, e)$
$x \xleftarrow{R} [\![1, q-1]\!]$
$(c, X) \leftarrow \mathcal{A}(k, \text{params}, xP_1, x^2P_1, \ldots, x^\ell P_1)$
Retourner 1 *si* $X = \dfrac{1}{x+c}P_0$ *et* 0 *sinon.*

On définit alors le succès de $\mathcal{A}$ pour résoudre le problème Diffie-Hellman flexible comme étant

$$\mathbf{Succ}^{\ell-\mathsf{sdh}}_{\mathsf{BDHGen},\mathcal{A}}(k) = \Pr\left[\mathbf{Exp}^{\ell-\mathsf{sdh}}_{\mathsf{BDHGen},\mathcal{A}}(k) = 1\right]$$

Étant donné $\tau \in \mathbb{N}$ et $\varepsilon \in [0,1]$, le problème ℓ-SDH est dit (k, τ, ε)-difficile s'il n'existe pas d'attaquant $\mathcal{A}$ fonctionnant en temps τ qui a un succès $\mathbf{Succ}^{\ell-\mathsf{sdh}}_{\mathsf{BDHGen},\mathcal{A}}(k) \geq \varepsilon$.

L'existence d'un couplage rend la version décisionnelle de ℓ-SDH facile. Cependant, il est tout à fait possible (et on verra qu'il est même opportun dans le cas des signatures indéniables que nous proposons dans la partie 2.5) de le considérer dans un contexte ou le problème Diffie-Hellman décisionnel est difficile, et donc pas forcément sur des groupes BDH. Ainsi, sa version décisionnelle est naturellement :

Problème ℓ-Diffie-Hellman flexible décisionnel (ℓ-DSDH) : soit $x \in [\![1, q-1]\!]$. Étant donné un entier ℓ, $(P, xP, x^2P, \ldots, x^\ell P) \in \mathbb{G}^{\ell+1}$, et un couple (Q, h) dans $\mathbb{G} \times [\![1, q-1]\!]$, décider si $Q = (x+h)^{-1}P$.

Définition 24 (ℓ-DSDH) *Soit k un entier, DLGen un générateur de groupe d'ordre premier et $\mathcal{A}$ un attaquant qui prend en entrée un paramètre de sécurité k, un triplet $(q, P, \mathbb{G})$ généré par DLGen et un $(\ell+3)$-uplet $(P, xP, x^2P, \ldots, x^\ell P, X, h) \in \mathbb{G}^{\ell+2} \times [\![1, q-1]\!]$, et qui retourne un bit $b \in \{0,1\}$. Considérons l'expérience aléatoire suivante, où $r \in \{0,1\}$:*

$$\boxed{\textit{Experiment } \mathbf{Exp}^{\ell-\mathsf{dsdh}-\mathsf{r}}_{\mathsf{DLGen},\mathcal{A}}(k)}$$
$(q, P, \mathbb{G}) \leftarrow \mathsf{DLGen}(k)$
$\text{params} \leftarrow (q, P, \mathbb{G})$
$x \xleftarrow{R} [\![1, q-1]\!]$, $h \xleftarrow{R} [\![1, q-1]\!]$
Pour $i = 1$ *à* ℓ *faire* $P_i = x^i P$
Si $r = 0$ *faire* $X \leftarrow (x+h)^{-1}P$
Si $r = 1$ *faire* $X \xleftarrow{R} \mathbb{G}$
$b \leftarrow \mathcal{A}^{\mathcal{O}_{\ell-SDH}}(k, \text{params}, P_1, \ldots, P_\ell, X, h)$
Retourner b.

On définit l'avantage de $\mathcal{A}$ pour résoudre le problème Diffie-Hellman décisionnel flexible comme étant

$$\mathbf{Adv}^{\ell-\mathsf{dsdh}}_{\mathsf{DLGen},\mathcal{A}}(k) = \left| Pr\left[\mathbf{Exp}^{\ell-\mathsf{dsdh}-1}_{\mathsf{DLGen},\mathcal{A}}(k) = 1\right] - Pr\left[\mathbf{Exp}^{\ell-\mathsf{dsdh}-0}_{\mathsf{DLGen},\mathcal{A}}(k) = 1\right]\right|.$$

Étant donné $\tau \in \mathbb{N}$ et $\varepsilon \in [0,1]$, le problème ℓ-DSDH est dit (k, τ, ε)-difficile s'il n'existe pas d'attaquant $\mathcal{A}$ fonctionnant en temps τ et ayant un avantage $\mathbf{Adv}^{\ell-\mathsf{dsdh}}_{\mathsf{DLGen},\mathcal{A}}(k) \geq \varepsilon$.

Sa version *gap* est nécessaire pour prouver l'inforgeabilité des signatures indéniables de la section 2.5 :

ℓ-Gap Diffie-Hellman flexible (ℓ-GSDH) : soit $x \in [\![1, q-1]\!]$. Étant donné un entier ℓ et $(P, xP, x^2P, \ldots, x^\ell P) \in \mathbb{G}^{\ell+1}$, calculer un couple $\big((x+h)^{-1}P, h\big)$ dans $\mathbb{G} \times [\![1, q-1]\!]$ pour un certain $h \in [\![1, q-1]\!]$, avec l'aide d'un oracle ℓ-DSDH.

Définition 25 (ℓ-GSDH) *Soit k un entier, DLGen un générateur de groupe d'ordre premier et $\mathcal{A}$ un attaquant qui prend en entrée un paramètre de sécurité k, un triplet $(q, P, \mathbb{G}) = \mathsf{DLGen}(k)$ et un $\ell+1$-uplet $(P, xP, x^2P, \ldots, x^\ell P) \in \mathbb{G}^{\ell+1}$. Il retourne un couple $(c, X) \in [\![1, q-1,]\!] \times \mathbb{G}$, en ayant accès à un oracle décisionnel pour le problème Diffie-Hellman flexible. Considérons l'expérience aléatoire suivante :*

$\boxed{\textit{Experiment } \mathbf{Exp}^{\ell-\mathsf{gsdh}}_{\mathsf{DLGen},\mathcal{A}}(k)}$
$(q, P, \mathbb{G}) \leftarrow \mathsf{DLGen}(k)$
$\text{params} \leftarrow (q, P, \mathbb{G})$
$x \xleftarrow{R} [\![1, q-1]\!]$
Pour $i = 1$ *à* ℓ *faire* $P_i = x^i P$
$(h, X) \leftarrow \mathcal{A}^{\mathcal{O}_{\ell-SDH}}(k, \text{params}, P_1, \ldots, P_\ell)$
Retourner 1 *si* $X = (x+h)^{-1}P$ *et* 0 *sinon.*

On définit le succès de $\mathcal{A}$ pour résoudre le problème Gap Diffie-Hellman flexible comme étant

$$\mathbf{Succ}^{\ell-\mathsf{gsdh}}_{\mathsf{DLGen},\mathcal{A}}(k) = \Pr\left[\mathbf{Exp}^{\ell-\mathsf{gsdh}}_{\mathsf{DLGen},\mathcal{A}}(k) = 1\right]$$

Étant donné $\tau \in \mathbb{N}$ et $\varepsilon \in [0, 1]$, le problème ℓ-GSDH est dit (k, τ, ε)-difficile s'il n'existe pas d'attaquant $\mathcal{A}$ qui, en temps τ, a un succès $\mathbf{Succ}^{\ell-\mathsf{gsdh}}_{\mathsf{DLGen},\mathcal{A}}(k) \geq \varepsilon$.

CCA Le problème précédent est inspiré de problèmes liés à la sécurité de schémas de *traitor tracing* [174, 238], et en particulier du problème k-CCA (pour *collusion attack algorithm with k traitors*) et du problème k-wDHA. Informellement, on peut les expliciter de la façon suivante :

Définition 26 (k-CAA) *Pour un entier k donné, $P \in \mathbb{G}$ d'ordre q, et $a \in [\![1, q-1]\!]$, le problème k-CAA (pour Collusion Attack Algorithm with k traitors) consiste, étant donné P, aP, k entiers $\{u_1, \ldots, u_k\}$ et k éléments $\left\{\frac{1}{u_i + a}P\right\}_{k \in [\![1,k]\!]}$ à calculer un point $\frac{1}{u_0 + a}P$, avec $u_0 \notin \{u_1, \ldots, u_k\}$.*

Définition 27 (k-wDHA) *Le problème k-wDHA (pour k-weak Diffie-Hellman Algorithm) consiste, étant donné $k+1$ points $P, xP, x^2P, \ldots, x^kP$, à calculer le point $x^{-1}P$.*

2

Des mathématiques pour la cryptologie

Sommaire

2.1 Les entiers **49**

2.1.1 Factorisation 50

2.1.2 RSA 50

2.1.3 Logarithme discret 52

2.2 Courbes elliptiques **53**

2.2.1 Définitions 53

2.2.2 Comptage des points d'une courbe elliptique 55

2.2.3 Logarithme discret sur une courbe elliptique 56

2.2.4 Remarques 57

2.3 Applications bilinéaires **58**

2.3.1 Exemples 59

Le terrain de jeu de la cryptographie moderne s'est longtemps cantonné aux anneaux d'entiers du type $\mathbb{Z}/N\mathbb{Z}$, ou dans le sous-groupe du groupe multiplicatif d'un corps fini. Même si, depuis, il s'est largement diversifié (corps de nombres, réseaux euclidiens, tores algébriques, tresses,...), RSA et les entiers restent privilégiés, avec un intérêt croissant pour les courbes elliptiques, introduites néanmoins en 1987, qui peuvent avantageusement remplacer les sous-groupes du groupe multiplicatif d'un corps fini, grâce à la difficulté de leur logarithme discret. Le petit séisme que représente en cryptographie l'introduction des applications bilinéaires les met aux places d'honneur pour succéder à RSA.

Ce sont donc aux candidats potentiels comme fonctions à sens unique que nous allons nous intéresser dans cette partie, en commençant par les fonctions issues des entiers, puis par celles venant de la géométrie algébrique.

2.1 Les entiers

Les deux principales fonctions considérées comme étant à sens unique sont la multiplication de grands nombres premiers et l'exponentiation discrète, que nous décrivons, ainsi que leur déclinaison, dans les sections suivantes.

2.1.1 Factorisation

Considérons la collection de fonctions suivantes :

$$\mathcal{M} = \{f_n : \{0,1\}^n \longrightarrow \{0,1\}^n, n \in \mathbb{N}\}$$

avec $f_n(r) = x$, où r est utilisé pour déterminer de façon unique deux nombres premiers p et q de taille $n/2$ (ou $(n+1)/2$) et x est calculé comme étant leur produit dans $\mathbb{N}$. Le produit de deux entiers peut être évidemment calculé en temps polynomial (en fait en temps $O(n \log n \log\log n)$). Le problème sous-jacent à l'inversion de l'une de ces fonctions est le problème de la factorisation.

Pour résoudre ce problème, il existe des méthodes "classiques" de complexité exponentielle : Fermat, ρ de Pollard, $p-1$, $p+1$, $\Phi_k(p)$ (où Φ_k est le k-ième polynôme cyclotomique),...

L'algorithme actuellement le plus performant est le *crible sur des corps de nombres* (*Number Field Sieve* - NFS) décrit dans [151, 152], et dont certaines variantes ont une complexité sous-exponentielle prouvée, sous certaines hypothèses :

$$e^{(c+o(1))(\log n)^{1/3}(\log\log n)^{2/3}},$$

où c est une constance proche de 2 ($c \approx 1,932$). Cet algorithme est l'attaquant de référence contre tout schéma basé sur la factorisation. Parmi les algorithmes probabilistes sous-exponentiels dont la complexité a été prouvée rigoureusement, on peut citer les raffinements de la méthode des carrés aléatoires de Dixon [77], par Vallée [239] puis Lenstra et Pomerance [154].

Le record de factorisation sur les challenges RSA a été annoncé le 9 mai 2005 par Bahr, Boehm, Franke et Kleinjung :

```
RSA-200 =
  27997833911221327870829467638722601621070446786955428537560009929326\
  12840010760934567105295536085606182235191095136578863710595448320\
  0657677509858055761357909873495014417886317894629518723786922182398
 =
  3532461934402770121272604978198464368671197400197\
  625023649303468776121253679423200058547956528088349
 ×
  7925869954478333033347085841480059687737975857364\
  219960734330341455767872818152135381409304740185467
```

2.1.2 RSA

RSA est la première incarnation d'un cryptosystème, et, sans doute à ce titre, ainsi qu'à celui de sa simplicité, il est l'algorithme le plus utilisé. Il est décrit, en chiffrement, dans la figure 2.1, et en signature dans la figure 2.2.

De façon informelle, le problème RSA est le suivant (il est formellement défini dans la partie précédente) :

Problème RSA [RSA] : soit p et q deux grands entiers premiers distincts de même taille. Étant donné $N = pq$ et $e \in \mathbb{Z}/N\mathbb{Z}^*$, et un élément $y \in \mathbb{Z}/N\mathbb{Z}^*$ quelconque, calculer la racine e-ième de y, c'est-à-dire $x \in \mathbb{Z}/N\mathbb{Z}^*$ tel que $y = x^e \mod N$.

May a montré dans [164] (voir aussi [163] pour des attaques sur RSA) que la connaissance d'un multiple de $\varphi(N)$ est équivalente à la connaissance de la factorisation de N, et il a exhibé une réduction polynomiale déterministe (jusqu'alors, la seule réduction connue était probabiliste).

Génération des clés : RSA.KeyGen

• k est un paramètre de sécurité,
• (N, e, d) est généré par $\mathsf{RSAGen}(k)$, (N, e) est la clé publique, d la clé secrète. Elles vérifient

$$ed \equiv 1 \mod \varphi(N).$$

Chiffrement : RSA.$\mathcal{E}$

• Étant donné un message m et la clé publique (N, e) de Bob, Alice calcule

$$c = m^e \mod N$$

Déchiffrement : RSA.$\mathcal{D}$

• Étant donné un chiffré c et sa clé secrète d, Bob calcule

$$c^d \mod N = m.$$

FIG. 2.1 – Chiffrement RSA "textbook"

La grande question ouverte à propos de RSA concerne l'équivalence entre le problème de la factorisation de modules RSA et le problème RSA lui-même. Boneh et Venkatesan ont suggéré dans [41] que le problème RSA avec un petit exposant semble plus facile que la factorisation.

De nombreux problèmes sont liés au problème RSA (ou à celui de la factorisation), notons en particulier le problème RSA flexible (voir [8]) :

Problème RSA flexible [FRSA] : soit p et q deux grands entiers premiers distincts de même taille. Étant donné $N = pq$ et un élément $y \in \mathbb{Z}/N\mathbb{Z}^*$ quelconque, calculer $(x, e) \in (\mathbb{Z}/N\mathbb{Z}^*)^2$, $e > 1$, tels que $y = x^e \mod N$.

Problème de la racine carrée [SQRT] : soit p et q deux grands entiers premiers distincts de même taille. Étant donné $N = pq$ et un élément $y \in \mathbb{Z}/N\mathbb{Z}^*$ quelconque, calculer une racine carrée, s'il en existe, de y, c'est-à-dire $x \in \mathbb{Z}/N\mathbb{Z}^*$ tel que $y = x^2 \mod N$.

Ce problème est en fait équivalent à celui de la factorisation, la réduction étant probabiliste.

Résiduosité quadratique [QR] : soit p et q deux grands entiers premiers distincts de même taille. Étant donné $N = pq$ et un élément $y \in \mathbb{Z}/N\mathbb{Z}^*$ de symbole de Jacobi $\left(\frac{y}{n}\right)$ égal à 1, déterminer s'il existe $s \in \mathbb{Z}/N\mathbb{Z}^*$ tel que $y = s^2 \mod n$.

Ce problème est une version décisionnelle du problème de la racine carrée. Pour une étude exhaustive de la cryptographie basée sur la résiduosité, on peut se référer à la thèse de Paillier [190].

L'utilisation de modules construits différemment, par exemple du type $N = p^2q$ (voir par exemple la thèse de Takagi [235]), ou $N = pqr$ (Multiprime de Compaq), ont été proposés pour améliorer les performances. L'exponentiation modulaire étant une opération relativement

Génération des clés : RSA.KeyGen

• k est un paramètre de sécurité,
• (N, e, d) est généré par RSAGen(k), (N, e) est la clé publique, d la clé secrète.
• Soit $[\{0,1\}^* \longrightarrow \mathbb{Z}/N\mathbb{Z}]$ une famille de fonctions de hachage h un élément pris au hasard dans cette famille.

Signature : RSA.$\mathcal{S}$

• Étant donné un message m et sa clé secrète d, Alice calcule

$$\sigma = h(m)^d \mod N$$

Vérification : RSA.$\mathcal{V}$

• Étant donné un couple (m, σ) et la clé publique (N, e), Bob accepte la signature si et seulement si

$$\sigma^e \mod N = h(m).$$

FIG. 2.2 – Signature RSA FDH

coûteuse (ce qui constitue un inconvénient important de RSA), de nombreux chercheurs ont tenté de la rendre plus efficace.

2.1.3 Logarithme discret

L'autre grande famille de problèmes sur lesquels reposent de nombreux cryptosystèmes est celle du logarithme discret.

Considérons un entier p premier, et un générateur g de $\mathbb{Z}/p\mathbb{Z}^*$. On peut considérer la collection de fonctions

$$EXP = \{f_n : \{0,1\}^n \longrightarrow \{0,1\}^n, n \in \mathbb{N}\}$$

avec $f_n(p, g, x) = (p, g, g^x)$. Le problème qui consiste à inverser EXP est le problème du calcul de logarithme discret.

Problème du logarithme discret modulo un premier [LOG$_p$] : soit p un grand entier premier et $g \in \mathbb{Z}/p\mathbb{Z}^*$ un générateur. Étant donné $y \in (\mathbb{Z}/p\mathbb{Z})^*$ quelconque, calculer $x \in [\![1, p-1]\!]$ tel que $y = g^x$.

En général, on se place dans un sous-groupe d'ordre premier :

Problème du logarithme discret dans un sous-groupe d'ordre premier [LOG$_q$] : soit p un grand entier premier, q un diviseur premier de $p-1$, et $g \in (\mathbb{Z}/p\mathbb{Z})^*$ d'ordre q. Étant donné $(y \in \mathbb{Z}/p\mathbb{Z})^*$, calculer un entier x, s'il existe, tel que $y = g^x \mod p$.

Problème du logarithme discret composite [LOG$_N$] : soit p et q deux grands entiers premiers. Étant donné $N = pq$, $g \in \mathbb{Z}/N\mathbb{Z}^*$ et $y \in \mathbb{Z}/N\mathbb{Z}^*$ quelconque, calculer un entier s, s'il existe, tel que $y = g^s \mod N$.

Ce problème est équivalent aux problèmes de la factorisation et du logarithme discret modulo un premier combinés : LOG_N est difficile si factoriser N est difficile, ou bien si le problème de logarithme discret modulo un premier est difficile.

Les algorithmes pour calculer des logarithmes discrets Il existe des algorithmes génériques pour calculer des logarithmes discrets dans n'importe quel groupe, qui sont de complexité exponentielle. On peut citer la méthode des *pas de bébé, pas de géant* de Shanks, la méthode ρ et celle des kangourous de Pollard (que l'on peut paralléliser). C'est cette dernière méthode, décrite par van Oorschot et Wiener [240], qui a permis de casser le challenge Certicom ECC22-97, un logarithme discret sur une courbe elliptique d'ordre un entier de 97 bits. Dans des groupes bien spécifiques, comme les corps finis, il existe des algorithmes de complexité sous-exponentielle : Adleman [2] a proposé un algorithme sous-exponentiel pour résoudre les logarithmes discrets dans $\mathbb{F}_p$ pour un premier p. Sa complexité est en $O(L_{1/2}(p)^2)$ en temps et $O(L_{1/2}(p))$ en espace, où $L_t(N) = \exp((\ln N)^t(\ln\ln N)^{(1-t)})$. Puis il l'a étendu avec De Marrais [3] à $\mathbb{F}_q$, où q n'est plus premier. Coppersmith, en 1984 [62], a donné un algorithme en $(L_{1/3}(q))^c$, pour une constante $c < 1,6$, valable seulement en caractéristique 2. Cet algorithme a permis a Thomé de résoudre un logarithme discret dans $\mathbb{F}_{2^{607}}$ (voir [236]). Le 18 juin 2005, Joux et Lercier ont annoncé un record de calcul de logarithme discret modulo un premier de 130 chiffres.

2.2 Courbes elliptiques

Miller à Crypto'85 [172] et Koblitz dans un article à Mathematics of Computation [139] en 1987 ont indépendamment proposé l'usage des courbes elliptiques comme un moyen alternatif pour réaliser des cryptosystèmes. Cet outil offre de façon avantageuse les conditions requises pour pouvoir définir des protocoles, à savoir une structure de groupe sur les points rationnels de la courbe avec une addition efficace, une large variété de courbes, et un problème du logarithme très difficile en général. En effet, sa difficulté est en général exponentielle, et des algorithmes sous-exponentiels (ou polynomiaux) ne sont connus que dans de rares cas (facilement identifiables). Ceci permet en particulier l'utilisation de clés plus courtes que dans le cas de la cryptographie sur les entiers. Le livre de Silverman [229] sur l'arithmétique des courbes elliptiques est une référence de choix pour étudier les courbes elliptiques, et leur utilisation en cryptographie est décrite et analysée dans les ouvrages de Blake, Seroussi et Smart [28], de Menezes [167], ou de Enge [85].

Dans la suite de cette partie, nous allons donner les informations nécessaires sur les courbes ellitiques pour justifier leur utilisation en cryptographie, en particulier montrer que l'on maîtrise relativement bien le groupe sous-jacent, et suggérer la difficulté du problème du logarithme discret.

2.2.1 Définitions

On considère un corps $\mathbb{K}$ et sa clôture algébrique $\overline{\mathbb{K}}$. On considère la relation $\sim$ définie sur $\mathbb{K}^3 \setminus \{(0,0,0)\}$ par

$$(x_1, y_1, z_1) \sim (x_2, y_2, z_2) \iff \text{il existe } \lambda \in \mathbb{K}^* \text{ tel que } x_1 = \lambda x_2,\ y_1 = \lambda y_2,\ z_1 = \lambda z_2.$$

Cette relation est une relation d'équivalence, et l'on note $(x : y : z)$ la classe d'équivalence contenant (x, y, z). Le plan projectif sur $\mathbb{K}$ est $P^2(\mathbb{K}) = \left(\mathbb{K}^3 \setminus \{(0,0,0)\}\right)/\sim$, l'ensemble de ces classes d'équivalence.

Une *équation de Weierstrass* est une équation homogène de degré 3 de la forme

$$Y^2Z + a_1XYZ + a_3YZ^2 = X^3 + a_2X^2Z + a_4XZ^2 + a_6Z^3,$$

avec $a_1, a_2, a_3, a_4, a_6 \in \overline{\mathbb{K}}$. L'équation de Weierstrass est dite *lisse* ou *non singulière* si pour tout $P = (x : y : z) \in P^2(\overline{\mathbb{K}})$ vérifiant $f(x, y, z) = y^3 + a_1xyz + a_3yz^2 - x^3 - a_2x^2z - a_4xz^2 - a_6z^3 = 0$, au moins l'une des 3 dérivées partielles $\partial f/\partial X$, $\partial f/\partial Y$, $\partial f/\partial Z$ ne s'annule pas en P. Si toutes ces dérivées s'annulent en un point P, ce dernier est appelé un *point singulier*, et l'équation de Weierstrass est dite *singulière*.

Une *courbe elliptique* E (ou courbe algébrique de genre 1) est l'ensemble de toutes les solutions dans $P^2(\overline{\mathbb{K}})$ d'une équation de Weierstrass non singulière. Il y a exactement 1 point de E dont la 3ème coordonnée est égale à 0 : c'est le point $(0 : 1 : 0)$. On appelle ce point *point à l'infini*, et on le note $\mathcal{O}$.

On écrit souvent l'équation de Weierstrass sous sa forme affine (en posant $x = X/Z$ et $y = Y/Z$) :

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6. \tag{2.1}$$

Une courbe elliptique est alors l'ensemble des solutions du plan affine $A^2(\overline{\mathbb{K}}) = \overline{\mathbb{K}} \times \overline{\mathbb{K}}$ de l'équation 2.1, avec le point à l'infini $\mathcal{O}$. Si $a_1, a_2, a_3, a_4, a_6 \in \mathbb{K}$, alors E est dite *définie* sur $\mathbb{K}$, et on la note $E/\mathbb{K}$. Si E est définie sur $\mathbb{K}$, on note l'ensemble des points $\mathbb{K}$-rationnels

$$E(\mathbb{K}) = \{(x, y) \in \mathbb{K}^2,\ y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6\} \cup \{\mathcal{O}\}.$$

Les deux résultats encourageants pour leur utilisation en cryptographie sont les suivants : tout d'abord, il existe une structure de groupe sur les points d'une courbe elliptique, et ce groupe a beaucoup de points lorsqu'on regarde la courbe sur un corps fini.

Ce premier fait, bien connu, se traduit géométriquement par la méthode dite *corde et tangente*, et s'exprime de la façon suivante : pour tout P et Q sur E

i) $\mathcal{O} + P = P$ et $P + \mathcal{O} = P$
ii) $-\mathcal{O} = \mathcal{O}$
iii) si $P = (x_1, y_1) \neq \mathcal{O}$, alors $-P = (x_1, -y_1 - a_1x_1 - a_3)$
iv) si $Q = -P$, alors $P + Q = \mathcal{O}$
v) si $P, Q \neq \mathcal{O}$ et $Q \neq -P$, et si l'on note R le troisième point d'intersection (en comptant la multiplicité, *cf* théorème de Bézout, et voir par exemple [121]) de la droite (PQ) si $P \neq Q$, ou de la tangente à E en P si $P = Q$ avec la courbe, alors $P + Q = -R$.

L'associativité, qui est la partie difficile à montrer, provient essentiellement du théorème de Riemann-Roch, voir [121, 229]. On a finalement :

Théorème 4 $(E, +)$ *est un groupe abélien d'élément neutre* $\mathcal{O}$. *Si* E *est définie sur* $\mathbb{K}$, *alors* $E(\mathbb{K})$ *est un sous-groupe de* E.

La structure du groupe utilisée en cryptographie est bien connue et donnée par le théorème suivant [85] :

Théorème 5 *Soit* $\mathbb{F}_q$ *le corps fini à* q *éléments, et* E *une courbe elliptique définie sur* $\mathbb{F}_q$. *Alors* $E(\mathbb{F}_q)$ *est un groupe abélien de rang 1 ou 2, et*

$$E(\mathbb{F}_q) \simeq \mathbb{Z}/n_1\mathbb{Z} \oplus \mathbb{Z}/n_2\mathbb{Z},$$

avec $n_1 \mid n_2$ *et* $n_1 \mid q - 1$.

L'addition sur la courbe s'écrit à l'aide de formules très simples, et les calculs peuvent être rendus très efficaces pour des applications cryptographiques dans des milieux contraints en terme de ressources de calculs (voir par exemple la thèse de Ciet [61]).

Le deuxième fait (le grand nombre de points) provient du théorème de Hasse. Considérons alors un corps $\mathbb{F}_q$ et le *Frobenius* φ défini par :

$$\begin{array}{rccc} \varphi : & E & \longrightarrow & E \\ & (x,y) & \longmapsto & (x^q, y^q) \\ & \mathcal{O} & \longmapsto & \mathcal{O} \end{array}$$

qui est un endomorphisme de groupe, et notons t la trace du Frobenius. Le cardinal du groupe $E(\mathbb{F}_q)$, noté $\#E(\mathbb{F}_q)$ peut être estimé par le résultat suivant :

Théorème 6 (Théorème de Hasse) *Soit E une courbe elliptique définie sur $\mathbb{F}_q$, alors*

$$\#E(\mathbb{F}_q) = q + 1 - t \text{ avec } |t| \leq 2\sqrt{q}.$$

Le théorème de Hasse a pour conséquence qu'il est possible de tirer des éléments dans le groupe $E(\mathbb{F}_q)$ avec distribution (presque) uniforme. Il suffit essentiellement de tirer aléatoirement un élément $\alpha \in \mathbb{F}_q$, et de résoudre l'équation quadratique en y obtenue en instantiant x par α dans l'équation 2.1. Il suffit alors de tirer à pile ou face quelle racine β choisir, s'il y a une solution, et de poser $P = (\alpha, \beta)$.

Pour les courbes sur $\mathbb{F}_p$, où p est premier, il existe une courbe elliptique dont le groupe de points rationnels $E(\mathbb{F}_p)$ a un ordre égal à n'importe quel entier dans $[\![p+1-2\sqrt{q}, p+1+2\sqrt{q}]\!]$.

***Remarque* 2 (Compression de points)** *Si l'on considère une courbe elliptique E définie sur un corps fini $\mathbb{F}_q$, alors un point affine $(x,y) \in E(\mathbb{F}_q)$ nécessite $2n$ bits pour être représenté, où $n = \log_2(q)$. On peut simplement réduire la taille de cette représentation à $n+1$ par la remarque suivante : connaissant x, la deuxième coordonnée y se retrouve simplement en résolvant une équation de degré 2, et il suffit donc d'un bit supplémentaire pour préciser laquelle des deux racines est la bonne. Cette astuce était déjà présente dans le papier de Miller [171].*

***Remarque* 3 (Autres courbes)** *D'autres objets issus de la géométrie algébrique peuvent remplacer les courbes elliptiques comme groupe de base. Les jacobiennes de courbes hyperelliptiques de rang supérieur ou égal à 2 ont été proposées par Koblitz [140] (les courbes elliptiques sont des courbes hyperelliptiques de rang 1, et sont isomorphes à leur jacobienne). Celles-ci ne sont intéressantes que pour des genres petits (2 ou 3). En effet, il existe des algorithmes meilleurs que les algorithmes génériques pour casser le logarithme discret dans les jacobiennes dès que le genre dépasse 4 (voir la thèse de Gaudry [98]). Les calculs dans les jacobiennes de courbes hyperelliptiques sont efficaces : voir la thèse de Lange [150]. Les jacobiennes de courbes non hyperelliptiques ont aussi été suggérées : un algorithme pour calculer dans la jacobienne d'une courbe de Picard est développé par Flon et Oyono dans [89]).*

2.2.2 Comptage des points d'une courbe elliptique

Avant d'utiliser les courbes elliptiques sur des corps finis, il est important de connaître exactement le cardinal du groupe des points (dont on a un ordre d'idée grâce au théorème de Hasse). Le problème du comptage de points sur une courbe est convenablement bien résolu d'un point de vue cryptographique (les records actuels de calcul de cardinalité dépassent largement les grandeurs cryptographiques). Deux grandes familles d'algorithmes (polynomiaux) sont utilisées : les algorithmes de types Schoof [217] (amélioré par Elkies et Atkin pour donner l'algorithme SEA), et les algorithmes de type Satoh [213]. Les différents algorithmes sont sensibles à la caractéristique du corps de base : les méthodes p-adiques (de type Satoh) ne sont efficaces qu'en petite caractérique. De nouvelles méthodes sophistiquées sont encore en cours de développement, qui sont des variantes, améliorations, extensions...

2.2.3 Logarithme discret sur une courbe elliptique

Les algorithmes génériques de calcul de logarithme discret s'appliquent évidemment aux courbes elliptiques.

Les courbes elliptiques sont particulièrement séduisantes de par l'absence, en général, d'un algorithme meilleur que ces algorithmes là, à savoir un algorithme sous-exponentiel pour y résoudre le problème du logarithme discret. L'algorithme ρ de Pollard, et ses variantes parallèles, ont une complexité en $O(\sqrt{p})$, où p est la taille du groupe.

Cependant, il existe un cas bien identifié de courbes elliptiques dont l'utilisation est définitivement proscrite : il s'agit des courbes *anomales*. Ces courbes, définies sur $\mathbb{F}_p$ ont leur trace de Frobenius égale à 1 (leur cardinal est divisible par p), et Araki et Satoh en 1998 [214], Semaev la même année [221] et Smart en 1999 [230] ont montré qu'on pouvait réduire le problème du logarithme discret sur ses courbes en un problème de logarithme discret dans le groupe $(\mathbb{Z}/p\mathbb{Z}, +)$.

Les courbes supersingulières sont un exemple de courbes elliptiques pour lesquelles il existe un algorithme sous-exponentiel pour résoudre le logarithme discret découvert en 1991 par Menezes, Okamoto et Vanstone [165], qui ont donc été écartées car considérées comme vulnérables, et qui ont été réhabilitées pour la raison même de leur condamnation : le couplage de Weil.

Théorème 7 *Soit E une courbe elliptique sur $\mathbb{F}_q$ de caractéristique p, les conditions suivantes sont équivalentes :*
i) l'anneau d'endomorphismes $\mathrm{End}(E)$ *est non-commutatif,*
ii) $E(\overline{\mathbb{F}_q})$ n'a pas de points d'ordre p,
iii) $p \mid t$, où t est la trace du Frobenius,
iv) il existe $k \in \mathbb{N}$ tel que $\varphi^k = \pm q^{k/2}$.

Définition 28 (Courbe elliptique supersingulière) *Une courbe elliptique E sur $\mathbb{F}_q$ vérifiant les conditions du théorème 7 est dite* supersingulière.

Une courbe non supersingulière est dite *ordinaire*.

Définition 29 (Degré MOV) *Soit $r \in \mathbb{N}$, on appelle* degré MOV *d'une courbe elliptique E sur $\mathbb{F}_q$ relativement à r le plus petit entier k tel que $E[r] = E(\mathbb{F}_{q^k})[r]$.*

Les courbes supersingulières présentent la particularité suivante, qui les a rendues célèbres, à savoir que leur degré MOV est petit. En effet, une courbe elliptique choisie au hasard a de fortes chance d'avoir un degré MOV arbitrairement grand.

Théorème 8 (Menezes, Okamoto, Vanstone) *Soit E sur $\mathbb{F}_q$ une courbe elliptique supersingulière et ℓ un diviseur premier $\#E(\mathbb{F}_q)$ ($\ell \neq p$ et $\ell \neq q-1$). Le degré MOV de E sur $\mathbb{F}_q$ relativement à ℓ est inférieur ou égal à 6.*

Il faut citer l'attaque par descente de Weil, suggérée par Frey en 1998 dans un exposé à ECC'98, et développée par Galbraith et Smart dans [97]. Cette attaque consiste à transporter le logarithme discret sur une courbe elliptique définie sur $\mathbb{F}_{p^r}$ (avec $r > 0$) sur une certaine jacobienne de courbe de genre[8] supérieure, et d'y appliquer un algorithme de calcul d'index. Cette attaque ne s'applique pas dans un cadre général.

En général, les attaques sur le logarithme discret sur les courbes elliptiques consistent à désigner une famille de courbes "faibles". Il est à noter un résultat récent de Gaudry [99], qui donne un algorithme de complexité meilleure que celle de l'agorithme ρ de Pollard pour *toutes* les courbes elliptiques définies sur une petite extension d'un corps fini $\mathbb{F}_p : \mathbb{F}_{p^r}$ avec $r \leq 5$. Cet algorithme peut être modifié pour obtenir une complexité sous-exponentielle (mauvaise) en $L_{3/4}(p)$.

Jusqu'à présent, une courbe elliptique quelconque définie sur un corps fini premier semble immune contre toute attaque.

2.2.4 Remarques

2.2.4.1 Suites de divisibilité elliptiques

Définition 30 *Une* suite de divisibilité elliptique *(EDS) est une suite d'entiers $(h_n)_{n\in\mathbb{N}}$ qui est une solution particulière de*

$$\omega_{m+n}\omega_{m-n} = \omega_{m+1}\omega_{m-1}\omega_n^2 - \omega_{n+1}\omega_{n-1}\omega_m^2 \tag{2.2}$$

et telle que

$$n \mid m \Longrightarrow h_n \mid h_m.$$

Shipsey a montré dans sa thèse [226] que de telles suites traduisent un certain comportement de points sur une courbe elliptique, comportement qui peut alors être décrit en utilisant des algorithmes simples manipulant les termes de ces suites. Le lien avec les courbes elliptiques provient du fait que les *polynômes de division* d'une courbe elliptique vérifient la relation de récurrence de la définition précédente. Les polynômes de division permettent de décrire le multiple d'un point : si $P = (x_0, y_0)$, alors

$$kP = \left(\frac{\phi_k(x_0,y_0)}{\psi_k^2(x_0,y_0)}, \frac{\omega_k(x_0,y_0)}{\psi_k^3(x_0,y_0)}\right)$$

où $\phi_k, \psi_k, \omega_k \in \mathbb{Z}[a_1,\ldots,a_6][x,y]$ sont définis par des relations de recurrence. En particulier, si E est une courbe elliptique sur $\mathbb{Q}$, et $P = (x,y) \in E(\mathbb{Q})$, alors

$$(\psi_k(x,y))_{n\in\mathbb{N}}$$

est une suite de divisibilité elliptique. Un entier m est dit *diviseur de la suite* $(h_n)_{n\in\mathbb{N}}$ s'il divise l'un des termes de la suite. Si $m \mid h_\rho$, mais $m \nmid h_r$ pour $r \mid \rho$, ρ est appelé le *rang d'apparition* de m dans $(h_n)_{n\in\mathbb{N}}$. Shipsey note

$$\langle h_n \rangle = \{h_{n-3}, h_{n-2}, h_{n-2}, h_{n-1}, h_n, h_{n+1}, h_{n+2}, h_{n+3}\}$$

et définit alors un problème de logarithme discret sur la suite :

[8] le genre d'une courbe correspond en quelque sorte à sa complexité

Problème du logarithme discret sur une EDS [EDSDL] : Étant donné une EDS $(h_k)_{k\in\mathbb{N}} \subset \mathbb{F}_p$ définie par h_1, h_2, h_3 avec un unique rang d'apparition pour p et $\langle h_n \rangle$, trouver k.

Après avoir proposé des algorithmes pour calculer les termes de la suite, Shipsey montre que le problème du logarithme discret sur une suite de divisibilité elliptique est plus facile que le logarithme discret sur une courbe elliptique, mais elle retrouve les attaques sur les courbes supersingulières et sur les courbes anomales en passant par les EDS. L'équivalence entre les deux problèmes n'a pas été démontrée. Il pourrait être intéressant de généraliser cette démarche sur des suites hyperelliptiques, pour lesquelles il existe un analogue des polynômes de division.

2.2.4.2 Autres applications cryptographiques

Les courbes elliptiques ont d'autres applications en cryptographie : on peut citer

- la construction de générateurs pseudo-aléatoires (voir la thèse Kaliski [132]) ;
- une méthode de factorisation (ECM) découverte par Lenstra en 1985 [153] qui a une complexité pour trouver un facteur p de l'entier à factoriser en $L_{1/2}(p)$. Cette méthode a permis en 1999 à Brent de factoriser le nombre de Fermat

$$\begin{aligned} F(11) = 2^{2^{11}} + 1 = 2^{2048} + 1 &= 319489 \times 974849 \times 167988556341760475137 \\ &\quad \times 3560841906445833920513 \times p_{564}. \end{aligned}$$

  L'entier p_{564} est un nombre premier de 564 décimales (prouvé premier par Morain) ;
- un test de primalité : les courbes elliptiques ont été proposées pour tester la primalité par Goldwasser et Kilian en 1986 [110], et implémentées par Atkin et Morain [6], sous le nom de ECPP. Cette méthode a permis de franchir en 2003 la "barrière mythique des 10000" chiffres décimaux grâce à l'implantation de J. Franke, T. Kleinjung et T. Wirth, en prouvant la primalité de $10^{9999} + 33603$.

2.3 Applications bilinéaires

Bien qu'utilisées en cryptographie en 1991 dans [165,166], les applications bilinéaires n'ont été considérées pour concevoir des protocoles que depuis que Joux, en 2000, a proposé un échange de clé tripartite en seulement un seul échange d'information [127,130]. En effet, Menezes, Okamoto et Vanstone ont introduit pour la première fois en 1991 le couplage de Weil pour transformer le problème du logarithme discret sur une courbe elliptique en un problème de logarithme discret dans le groupe multiplicatif d'un corps fini. Ce résultat constitue la première solution sous-exponentielle au problème du logarithme discret sur une courbe elliptique, mais elle ne fonctionne essentiellement que sur les courbes supersingulières, ainsi que sur certaines courbes ordinaires (voir partie 2.2.3). En 1994, Frey et Rück [91] ont généralisé cette attaque en utilisant le couplage de Tate. C'est l'existence même de ces couplages sur les courbes elliptiques qui rendait l'attaque effective. Depuis 2000, les applications bilinéaires sont abordées indépendamment du support ; le fait est que ces deux couplages (Weil et Tate) sont les seuls connus permettant d'instancier tous les cryptosystèmes qui utilisent des applications bilinéaires. Celles-ci ont focalisé une partie de la communauté scientifique et permis de spécifier un grand nombre de nouveaux cryptosystèmes jouissant de propriétés que l'on n'avait pu atteindre jusqu'alors, ou bien de manière inefficace. Ainsi, en 2001, Boneh et Franklin [35,36] ont construit le premier algorithme de chiffrement basé sur l'identité efficace, un des graals des cryptologues depuis l'introduction de la cryptographie basée sur l'identité en 1984 par A. Shamir [223].

2.3.1 Exemples

Il existe essentiellement deux types d'application bilinéaire admissible : les couplages issus de la géométrie algébrique, et l'exponentiation discrète, celle-ci s'avérant être le mauvais élève. Dans la suite de cette section, nous passons en revue ces deux candidats.

2.3.1.1 Couplages de Weil et Tate

Nous allons donner ici un aperçu des applications bilinéaires admissibles offertes par la géométrie algébrique et actuellement utilisées en cryptographie, à savoir les couplages de Weil et de Tate. Cette présentation s'inspire du livre de Menezes [167] et du manuscript de Dupont [81].

Avant de présenter ces couplages, il est nécessaire d'introduire quelques notions supplémentaires sur les courbes elliptiques. On note $[n]$ la multiplication scalaire par un entier n , qui est un endormorphisme de $E(\overline{\mathbb{F}_q})$. On note $E[n]$ le noyau de $[n]$: c'est un sous-groupe de $E(\overline{\mathbb{F}_q})$ appelé le *groupe des points de n-torsion.* Il constitue donc l'ensemble des points de la courbe qui sont "tués" par n. On note $E(\mathbb{K})[n] = E[n] \cap E(\mathbb{K})$ pour tout $\mathbb{F}_q \subset \mathbb{K} \subset \overline{\mathbb{F}_q}$. Sa structure est décrite dans la proposition suivante :

Proposition 1 (Structure du groupe de torsion) *Soit E une courbe elliptique sur $\mathbb{F}_q$ de caractéristique p, et $m \in \mathbb{N}^*$.*
- *Si* $\mathrm{pgcd}(p, m) = 1$, *alors* $E[m] \simeq \mathbb{Z}/m\mathbb{Z} \oplus \mathbb{Z}/m\mathbb{Z}$.
- *Si $m = p^\ell$, pour un $\ell \in \mathbb{N}$, alors*
  - *$E[m] \simeq \{\mathcal{O}\}$ si E est supersingulière*
  - *$E[m] \simeq \mathbb{Z}/p^\ell\mathbb{Z}$ sinon.*

La notion de *diviseur* sur une courbe elliptique est aussi nécessaire pour définir les couplages de Weil et de Tate. Un diviseur de $E(\mathbb{F}_q)$ est une somme formelle finie de points de $E(\mathbb{F}_q)$, qu'on écrit :

$$D = \sum_{P \in E(\mathbb{F}_q)} n_P(P),$$

où les $n_P \in \mathbb{Z}$ sont presque tous nuls. On note $\mathrm{Div}_{\mathbb{F}_q}(E)$ le groupe des diviseurs de $E(\mathbb{F}_q)$ (c'est le groupe libre engendré par les points de la courbe). On définit le *support* d'un diviseur D comme étant

$$\mathrm{supp}(D) = \{P \mid n_P \neq 0\}$$

et son *degré* comme étant

$$\deg D = \sum_{P \in E(\mathbb{F}_q)} n_P.$$

On note $\mathrm{Div}^0_{\mathbb{F}_q}(E)$ le sous groupe de $\mathrm{Div}_{\mathbb{F}_q}(E)$ des diviseurs de degré 0. Si K est le corps des fonctions de la courbe (c'est-à-dire le corps des fractions de l'anneau $\mathbb{F}_q[x, y]/(f(x, y))$, où $f(x, y)$ est une équation affine de E), et noté $\mathbb{F}_q(E)$, le diviseur d'une fonction f de K est

$$\mathrm{div}(f) = \sum_{P \in E(\mathbb{F}_q)} \mathrm{ord}_P(f)(P),$$

où ord est l'ordre de f en P. Un diviseur D de $\mathrm{Div}^0_{\mathbb{F}_q}(E)$ est dit *principal* si D est le diviseur d'une fonction de E, c'est-à-dire qu'il existe $f \in \overline{\mathbb{F}_q}(E)$ telle que $D = \mathrm{div}(f)$. On caractérise aisément un diviseur principal, puisque $D = \sum_{P \in E(\mathbb{F}_q)} n_P(P)$ est principal si et seulement si (voir [229])

$$\deg(D) = 0 \text{ et } \sum_{P \in E(\mathbb{F}_q)} n_P P = \mathcal{O}.$$

On note $\mathrm{Princ}_{\mathbb{F}_q}(E)$ le sous groupe de $\mathrm{Div}^0_{\mathbb{F}_q}(E)$ formé par les diviseurs principaux. On peut alors définir la relation d'équivalence $\sim$ entre des diviseurs de degré 0 de la façon suivante. Soit D_1 et D_2 dans $\mathrm{Div}^0_{\mathbb{F}_q}(E)$, ils sont dits *équivalents*, et l'on note $D_1 \sim D_2$ si et seulement si $D_1 - D_2 \in \mathrm{Princ}_{\mathbb{F}_q}(E)$, c'est-à-dire que $D_1 = D_2 + \mathrm{div}(f)$ pour une certaine fonction $f \in \overline{\mathbb{F}_q}(E)$. Le groupe quotient $\mathrm{Div}^0_{\mathbb{F}_q}(E)/\mathrm{Princ}_{\mathbb{F}_q}(E)$ est appelé le *groupe de Picard de E*.

Soit $D = \sum_{P \in E(\mathbb{F}_q)} n_P(P) \in \mathrm{Div}_{\mathbb{F}_q}(E)$ un diviseur et $f \in \overline{\mathbb{F}_q}(E)^*$ une fonction, tels que $\mathrm{div}(f)$ et D ont des supports disjoints, alors il est possible de définir l'évaluation de f en D par

$$f(D) = \prod_{P \in \ \mathrm{supp}(D)} f(P)^{n_P}.$$

Définitions Soit $\mathbb{F}_q$ le corps fini à q éléments de caractérisique p, et soit $m \in \mathbb{N}$ premier à p. On note $\mu_m \subset \overline{\mathbb{F}_q}^*$ le groupe des racines m-ièmes de l'unité. On peut alors définir le couplage de Weil :

Définition 31 (Couplage de Weil - [244]) *Soit P et Q dans $E[m]$, et A et B deux diviseurs de degré 0 tels que $A \sim (P) - (\mathcal{O})$ et $B \sim (Q) - (\mathcal{O})$, avec $\mathrm{supp}(A) \cap \mathrm{supp}(B) = \emptyset$. Soit f_A et f_B dans $\overline{\mathbb{F}_q}(E)$ telles que $\mathrm{div}(f_A) = mA$ et $\mathrm{div}(f_B) = mB$. Le* couplage de Weil e_m *est la fonction*

$$\begin{array}{rccl} e_m : & E[m] \times E[m] & \longrightarrow & \mu_m \\ & (P, Q) & \longmapsto & e_m(P, Q) = f_A(B)/f_B(A). \end{array}$$

Balasubramanian et Koblitz dans [7] ont montré que, si m est un diviseur de $\#E(\mathbb{F}_q)$, on a :

Théorème 9 *Si $m \neq p$, et $m \nmid q - 1$, alors :*

$$E[m] = E(\mathbb{F}_{q^k})[m] \Longleftrightarrow m \mid q^k - 1.$$

Ainsi, si m vérifie les hypothèses précédentes, l'ordre de q modulo m est justement le degré MOV de $E(\mathbb{F}_q)$ relativement à m.

On note ℓ l'ordre de q modulo m, et l'on suppose de plus que m divise $\#E(\mathbb{F}_{q^\ell})$, de sorte que $E(\mathbb{F}_{q^\ell})$ contienne au moins m points de m-torsion. On a alors $\mu_m \subset \mathbb{F}^*_{q^\ell}$. Le couplage de Tate, lui, se définit ainsi :

Définition 32 (Couplage de Tate) *Soit $P \in E(\mathbb{F}_{q^\ell})[m]$ et $Q \in E(\mathbb{F}_{q^\ell})$. Soit $f_P \in \mathbb{F}_{q^\ell}(E)$ telle que $\mathrm{div}(f_P) = m(P) - m(\mathcal{O})$ et $D \in Div_{\mathbb{F}_{q^\ell}}(E)$ tel que $D \sim (Q) - (\mathcal{O})$. Le* couplage de Tate *$t_m$ est la fonction*

$$\begin{array}{rccl} t_m : & E(\mathbb{F}_{q^\ell})[m] \times E(\mathbb{F}_{q^\ell})/[m]E(\mathbb{F}_{q^\ell}) & \longrightarrow & \mathbb{F}^*_{q^\ell}/(\mathbb{F}^*_{q^\ell})^m \\ & (P, Q) & \longmapsto & f_m(P, Q) = f_P(D). \end{array}$$

Propriétés Les deux paragraphes suivants montrent que les couplages de Tate et Weil sont bien les applications bilinéaires que l'on recherche.

Proposition 2 ([229]) *Le couplage de Weil satisfait les propriétés suivantes :*
- *identité :* $\forall P \in E[m]$, $e_m(P,P) = 1$;
- *bilinéarité :* $\forall P, Q, R \in E[m]$, $e_m(P+Q, R) = e_m(P,R)e_m(Q,R)$ *et* $e_m(P, Q+R) = e_m(P,Q)e_m(P,R)$;
- e_m *est alterné :* $\forall P, Q \in E[m]$, $e_m(P,Q) = e_m(Q,P)^{-1}$;
- *non-dégénéré : si* $P \in E[m]$, *alors* $e_m(P, \mathcal{O}) = 1$. *De plus, si* $e_m(P,Q) = 1$ $\forall Q \in E[m]$, *alors* $P = \mathcal{O}$.
- *si* $E[m] \subset E(\mathbb{F}_q)$, *alors* $e_m(P,Q) \in \mathbb{F}_q$, $\forall P, Q \in E[m]$;
- *compatibilité : si* $P \in E[m]$ *et* $Q \in E[mm']$, *alors* $e_{mm'}(P,Q) = e_m(P, m'Q)$.

Proposition 3 ([91]) *Le couplage de Tate est bilinéaire et non-dégénéré.*

Le problème essentiel du couplage de Weil provient du fait que pour tout $P \in E[m]$, $e_m(P,P) = 1$. Il est cependant possible de modifier le couplage de Weil pour faire disparaître cette propriété, au moins sur les courbes supersingulières. Considérons donc une courbe supersingulière de degré MOV k relativement à un diviseur premier m de $\#E(\mathbb{F}_q)$. Lorsqu'il existe (et ce n'est pas toujours le cas) un morphisme

$$\Psi : E(\mathbb{F}_{q^k}) \longleftrightarrow E(\mathbb{F}_{q^k})$$

non $\mathbb{F}_q$-rationnel, si P désigne un générateur de $E(\mathbb{F}_q)[m]$, P et $\Psi(P)$ engendrent $E[m]$. On peut alors définir le couplage de Weil modifié :

$$\begin{array}{rccl} \hat{e}_m : & E[m] \times E[m] & \longrightarrow & \mathbb{F}_{q^k}^* \\ & (R,S) & \longmapsto & \hat{e}_m(R,S) = e_m(R, \Psi(S)). \end{array}$$

Algorithme de Miller Ces applications semblent donc bien être de bonnes candidates à des applications bilinéaires admissibles. . . si toutefois on peut les calculer.

En 1986, Miller, dans [171] propose un algorithme polynomial pour calculer les couplages de Weil et Tate.

L'algorithme de la figure 2.3 calcule le couplage de Tate des points $P \in E(\mathbb{F}_{q^k})[n]$ et $Q \in E(\mathbb{F}_{q^k})$. Si U et V sont deux points de $E(\mathbb{F}_{q^k})$, on note $g_{U,V} \in \mathbb{F}_{q^k}(E)$ la droite passant par U et V donnée par $l_1 y + l_2 x + l_3 = 0$ et $g_{U,-U}$ est notée g_U.

La complexité de cet algorithme est en $O(\log n \cdot M(q,k))$, où $M(q,k)$ est la complexité de la multiplication dans $\mathbb{F}_{q^k}$. Si k, le degré MOV, est trop élevé (ce qui est le cas en général pour une courbe quelconque), les calculs ne seront pas faisables. Il est donc important de trouver des courbes de petit degré MOV.

***Remarque* 4 (Autres courbes)** *Outre les coubres supersingulières, ce sont les courbes elliptiques dites MNT qui sont des candidates potentielles pour implanter les algorithmes de couplages. En 2001, Miyaji, Nakabayashi et Takano [175] ont décrit une construction de courbes d'ordre premier de degré MOV petit (3, 4, ou 6). Ces résultats ont été étendus par exemple par Scott et Barreto [219]. Dupont, Enge et Morain [82] ont aussi proposé une construction de courbes elliptiques*

ENTRÉE : $P \in E(\mathbb{F}_{q^k})[m]$ et $Q \in E(\mathbb{F}_{q^k})$
SORTIE : $t_m(P,Q)$

1. Tirer au hasard $Q' \in E(\mathbb{F}_{q^k})$ et calculer $S = Q + Q' \in E(\mathbb{F}_{q^k})$
2. Poser $t = \lfloor \log_2(m) \rfloor$ et écrire $m = \sum_{i=1}^{t} m_i 2^i$
   Poser $f = 1$ et $V = P$
3. Pour $i = t-1$ à 0 faire
   - Poser $f = f^2(g_{V,V}(S)g_{2V}(Q'))/(g_{2V}(S)g_{V,V}(Q'))$ et $V = 2V$
   - si $m_i = 1$ poser $f = f(g_{V,P}(S)g_{V+P}(Q'))/g_{V+P}(S)g_{V,P}(Q'))$ et $V = V + P$
4. Retourner f

FIG. 2.3 – Algorithme de Miller pour le couplage de Tate

ordinaires de petit degré MOV. Galbraith a étudié les courbes supersingulières de genre supérieur dans [94], et Barreto, Galbraith, O'hEigeartaigh et M. Scott [9] ont proposé un algorithme efficace de calcul de couplage pour des courbes elliptiques et hyperelliptiques supersingulières en caractéristique 2.

***Remarque* 5 (Survol des applications)** *Les couplages de Weil et Tate sont les briques de base d'une multitude de cryptosystèmes [128]. Il est difficile d'être exhaustif, mais on peut citer comme applications les versions "basées sur l'identité" de toutes les signatures possibles, de tous les types de chiffrement, la distribution de clés non interactive, les signatures courtes, et les amélorations d'un certain nombre de cryptosytèmes existants...*

Le couplage de Tate semble être le meilleur des deux candidats, en termes de coût de calcul (deux fois moins cher que le couplage de Weil), et peut être utilisé sur des courbes sur lesquelles Weil ne peut pas l'être. Les questions à se poser lorsqu'on veut utiliser l'un de ces objets concernent la caractéristique du corps de base (2, 3, p ?), le degré MOV, la courbe (supersingulière, MNT ?). De nombreux progrès sont faits en matière de calcul de ces couplages. [218]

2.3.1.2 Exponentiation discrète

Le cas de l'exponentiation discrète comme application bilinéaire est particulier pour plusieurs raisons. Si l'on considère un groupe $(\mathbb{G}, +)$ d'ordre un premier q et l'exponentiation discrète

$$\begin{array}{rccc} \exp: & \mathbb{G} \times \mathbb{Z}/q\mathbb{Z} & \longrightarrow & \mathbb{G} \\ & (P, a) & \longmapsto & aP, \end{array}$$

celle-ci est effectivement une application bilinéaire admissible. Contrairement aux couplages précédents, le morphisme $\mathbb{G} \longrightarrow \mathbb{Z}/q\mathbb{Z}$ n'est pas calculable, et pour cause, il s'agit du logarithme discret. En outre, le problème du logarithme discret dans $(\mathbb{Z}/q\mathbb{Z}, +)$ est facile. Ces deux faits rendent en général l'utilisation de l'exponentiation discrète pour instancier une application bilinéaire dans un protocole inadaptée (en fait dangereuse).

3

Retour sur le chiffrement et la signature

Sommaire

3.1 Chiffrement . . . 63
3.1.1 Sécurité sémantique - IND-CCA . . . 64
3.1.2 Indistinguabilité des clés : Key-Privacy . . . 65
3.2 Signature . . . 66
3.2.1 Modèle formel . . . 66
3.2.2 Diverses signatures utiles par la suite . . . 67
3.3 Remarques . . . 72
3.3.1 Enregistrement des clés des utilisateurs . . . 72
3.3.2 Preuves de sécurité . . . 72

Nous revenons dans cette partie sur des notions liées au chiffrement et aux signatures qui nous seront utiles par la suite. Cela concerne les définitions formelles de la sécurité sémantique d'un chiffrement ainsi que celle de *key-privacy*. Puis nous donnerons le modèle formel de l'inforgeabilité d'un schéma de signature, et décrirons des schémas de signatures qui nous serviront pour en construire de nouveaux.

3.1 Chiffrement

On note $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathcal{E}, \mathcal{D})$ un schéma de chiffrement. La notion de *sens-unique* sous-entend que tout adversaire $\mathcal{A}$ essayant d'inverser $\mathcal{E}$ sans la clé secrète a une probabilité de succès négligeable sur l'espace $\mathcal{M} \times \Omega$ où $\mathcal{M}$ est l'espace des messages et Ω l'espace des aléas utilisés dans le schéma de chiffrement et des aléas internes à $\mathcal{A}$. Le succès de $\mathcal{A}$ est défini en considérant l'expérience aléatoire suivante :

$\boxed{\text{Experiment } \mathbf{Exp}^{\mathsf{ow}}_{\Pi,\mathcal{A}}(k)}$
$\text{params} \leftarrow \Pi.\mathsf{Setup}(k)$
$(\text{pk}, \text{sk}) \leftarrow \Pi.\mathsf{KeyGen}(k, \text{params})$
$m \stackrel{R}{\leftarrow} \mathcal{M}$
$c \leftarrow \Pi.\mathcal{E}(m, r)$
$m^\star \leftarrow \mathcal{A}(k, \text{params}, pk, c)$
$d \leftarrow 1$ si $m^\star = m$ et 0 sinon
Retourner d

Le succès de $\mathcal{A}$ est alors défini par

$$\mathbf{Succ}^{\mathsf{ow}}_{\Pi,\mathcal{A}}(k) = \Pr\left[\mathbf{Exp}^{\mathsf{ow}}_{\Pi,\mathcal{A}}(k) = 1\right].$$

Cette notion, comme nous l'avons déjà souligné, n'est pas suffisante pour caractériser un bon système de chiffrement. C'est la notion de sécurité sémantique, ou d'indistinguabilité, qui est la notion de sécurité requise.

3.1.1 Sécurité sémantique - IND-CCA

Nous donnons donc ici la définition formelle de la sécurité sémantique d'un schéma de chiffrement, qui correspond au fait qu'il est calculatoirement impossible pour un attaquant de distinguer lequel de deux messages, qu'il a lui-même choisis, a été chiffré, avec une probabilité suffisamment éloignée de $1/2$.

3.1.1.1 Définition

L'attaquant que l'on considère dans les attaques contre la sécurité sémantique fonctionne en deux étapes. Il a accès durant son attaque à un oracle de *déchiffrement*[9]. Lorsque l'accès à cet oracle n'est autorisé que dans la première étape, c'est-à-dire *avant* la connaissance du challenge, l'attaque est dite *non-adaptative*, et on parle de IND-CCA1. Lorsque l'accès est illimité l'attaque est dite *adaptative* : c'est la notion de IND-CCA2. Dans ce dernier cas, l'attaquant ne doit pas interroger l'oracle sur le challenge. C'est ce dernier scénario qui nous intéresse. L'attaquant fonctionne donc de la façon suivante : dans sa première étape find, il prend en entrée un paramètre de sécurité, des paramètres publics et une clé publique cible. À la fin de cette étape, il propose à son challenger un couple de messages (m_0, m_1) (et sort une information d'état). Dans sa deuxième étape guess, il reçoit le chiffré de l'un des deux messages (ainsi que son information d'état), et sort un bit. Il réussit son attaque si ce bit correspond au message qui a été chiffré.

Définition 33 (IND-CCA2) *Soit k un entier, $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathcal{E}, \mathcal{D})$ un schéma de chiffrement et $\mathcal{A}$ un adversaire IND-CCA2 contre Π. Considérons les expériences aléatoires suivantes, où k est un paramètre de sécurité et $r \in \{0,1\}$:*

$$\boxed{\textit{Experiment } \mathbf{Exp}^{\mathsf{ind-cca2}-r}_{\Pi,\mathcal{A}}(k)}$$

$$\begin{array}{l}
\text{params} \leftarrow \Pi.\mathsf{Setup}(k) \\
(pk, sk) \leftarrow \Pi.\mathsf{KeyGen}(k, \text{params}) \\
(m_0, m_1, \mathcal{I}) \leftarrow \mathcal{A}^{\mathcal{D}_{sk}}(k, \mathsf{find}, \text{params}, pk) \\
y \leftarrow \mathcal{E}_{pk}(m_r) \\
d \leftarrow \mathcal{A}^{\mathcal{D}_{sk}}(k, \mathsf{guess}, \mathcal{I}, \text{params}, pk, y) \\
\textit{Retourner } d
\end{array}$$

*On définit alors l'*avantage *de l'adversaire $\mathcal{A}$, comme étant*

$$\mathbf{Adv}^{\mathsf{ind-cca2}}_{\Pi,\mathcal{A}}(k) = \left|\Pr\left[\mathbf{Exp}^{\mathsf{ind-cca2}-1}_{\Pi,\mathcal{A}}(k) = 1\right] - \Pr\left[\mathbf{Exp}^{\mathsf{ind-cca2}-0}_{\Pi,\mathcal{A}}(k) = 1\right]\right|.$$

Étant donné $\tau \in \mathbb{N}$ et $\varepsilon \in [0,1]$, le schéma Π est dit (k, τ, ε)-IND-CCA2, si aucun adversaire IND-CCA2 $\mathcal{A}$ fonctionnant en temps τ, n'a un avantage $\mathbf{Adv}^{\mathsf{ind-cca2}}_{\Pi,\mathcal{A}}(k) \geq \varepsilon$.

[9] il pourrait avoir accès à des oracles moins puissants, comme un oracle qui vérifie qu'un chiffré est valide ou non, ou un oracle qui répond si un couple (m, c) est un couple formé d'un message et de l'un de ses chiffrés

Les exemples les plus connus de chiffrement IND-CCA2 sont le chiffrement de Cramer et Shoup [65], qui est IND-CCA2 dans le modèle standard, sous DDH et la résistance à la seconde préimage de la fonction de hachage et RSA-OAEP [24,93,228], qui est IND-CCA2 dans le modèle de l'oracle aléatoire sous l'hypothèse que RSA est à sens unique. On peut aussi citer REACT, procédé générique, qui convertit une fonction à sens unique à trappe en un chiffrement IND-CCA2 [189].

La figure[10] 3.1 récapitule et compare les différentes notions de sécurité pour le chiffrement, où

- NM signifie non-malléabilité (*non malleability*)
- IND signifie indistinguabilité (*indistinguishability*)
- CCA signifie attaque à chiffrés choisis (*chosen ciphertext attack*)
  - 1 pour non adaptative
  - 2 pour adaptative
- CPA signifie attaque à clairs choisis (*chosen plaintext attack*)

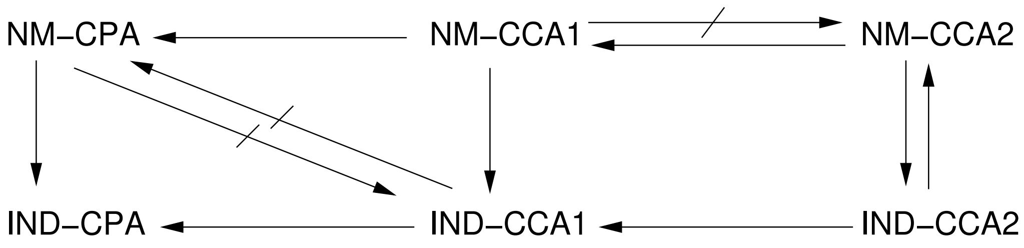


Fig. 3.1 – Relations entre les différentes notions de sécurité du chiffrement [16]

3.1.2 Indistinguabilité des clés : Key-Privacy

La notion d'indistinguabilité des clés est une notion d'anonymat pour le chiffrement, introduite par Bellare, Boldyreva, Desai et Pointcheval dans [14]. Elle traduit le fait qu'un attaquant en possession d'un ensemble de clés n'est pas capable de déterminer laquelle d'entres elles a été utilisée pour produire un chiffré.

3.1.2.1 Définition

L'attaquant contre l'indistinguabilité des clés, dans une attaque à chiffrés choisis adaptative (IK-CCA) fonctionne en deux étapes : dans la première, il choisit le message qui va être chiffré grâce à l'une des deux clés publiques qu'il a à sa disposition. Dans sa deuxième phase, il essaye de deviner laquelle a réellement servi pour chiffrer.

Définition 34 (Indistinguabilité des clés [14]) *Soit k un entier, $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathcal{E}, \mathcal{D})$ un schéma de chiffrement et $\mathcal{A}$ un adversaire IK-CCA contre Π. Considérons les expériences aléatoires suivantes, où k est un paramètre de sécurité et $r \in \{0,1\}$:*

[10] à laquelle est associée un étrange copyright *cf* [201] Fig. 2.7 p. 71

$$\boxed{\textit{Experiment } \mathbf{Exp}^{\mathsf{ik-cca}-r}_{\Pi,\mathcal{A}}(k)}$$
$$\begin{array}{l}
\text{params} \leftarrow \Pi.\textsf{\textit{Setup}}(k) \\
(pk_0, sk_0) \leftarrow \Pi.\textsf{\textit{KeyGen}}(k, \text{params}) \\
(pk_1, sk_1) \leftarrow \Pi.\textsf{\textit{KeyGen}}(k, \text{params}) \\
(m^\star, \mathcal{I}) \leftarrow \mathcal{A}^{\mathcal{D}_{sk_0}, \mathcal{D}_{sk_1}}(k, \textsf{\textit{find}}, \text{params}, pk_0, pk_1) \\
y \leftarrow \mathcal{E}_{pk_r}(m^\star) \\
d \leftarrow \mathcal{A}^{\mathcal{D}_{sk_0}, \mathcal{D}_{sk_1}}(k, \textsf{\textit{guess}}, \mathcal{I}, \text{params}, pk_0, pk_1, y) \\
\textit{Retourner } d
\end{array}$$

*On définit alors l'*avantage *de l'adversaire $\mathcal{A}$, comme étant*

$$\mathbf{Adv}^{\mathsf{ik-cca}}_{\Pi,\mathcal{A}}(k) = \left| \Pr\left[\mathbf{Exp}^{\mathsf{ik-cca}-1}_{\Pi,\mathcal{A}}(k) = 1\right] - \Pr\left[\mathbf{Exp}^{\mathsf{ik-cca}-0}_{\Pi,\mathcal{A}}(k) = 1\right] \right|.$$

Étant donné $\tau \in \mathbb{N}$ et $\varepsilon \in [0,1]$, le schéma Π est dit (k, τ, ε)-IK-CCA si aucun adversaire IK-CCA $\mathcal{A}$ fonctionnant en temps τ, n'a un avantage $\mathbf{Adv}^{\mathsf{ik-cca}}_{\Pi,\mathcal{A}}(k) \geq \varepsilon$.

Dans [14], Bellare *et al.* montrent que le chiffrement El Gamal [84] possède l'anonymat des clés sous une attaque à clairs choisis, sous l'hypothèse que DDH est difficile. Ils montrent également que le chiffrement de Cramer et Shoup [65] satisfait cette propriété d'anonymat sous une attaque à chiffrés choisis adaptative, aussi sous DDH. Ils proposent en outre une variante de RSA-OAEP qui fournit l'anonymat dans le modèle de l'oracle aléatoire, si RSA est une fonction à sens-unique.

3.2 Signature

Après avoir précisé le modèle de sécurité d'un schéma de signatures, nous décrivons quelques constructions utiles de signatures “spéciales”. Pour chaque famille de signatures, possédant des propriétés particulières, il est important de reprendre le modèle de sécurité depuis le début en prenant soin de définir les buts et les moyens de l'attaquant. De nombreux schémas proposés sans une présentation précise du modèle de sécurité se sont avérés non sûrs. Aussi commençons-nous par décrire le modèle de sécurité des signatures numériques classiques, qui servira de base à tous les schémas. En effet, le modèle d'attaquant pour tous les schémas de signatures “spéciales” est une déclinaison de celui-ci, obtenue en tenant compte des informations supplémentaires à la disposition de l'attaquant.

3.2.1 Modèle formel

La notion de sécurité admise pour les schémas de sécurité est la résistance à une forge existentielle sous une attaque à messages choisis (*existential unforgeability against a chosen message attack* - EF-CMA), telle qu'elle a été définie par Goldwasser, Micali et Rivest en 1988 dans [116]. L'attaquant EF-CMA contre le schéma de signatures a accès à un oracle de signatures Σ qu'il peut interroger sur tous les messages de son choix, avec la condition que la signature dans le couple message/signature qu'il renvoie à la fin de son attaque ne provient pas de l'oracle de signature. Dans le cas de signatures probabilistes, l'attaquant peut cependant interroger l'oracle de signature sur le message qu'il essaye d'attaquer. Sa tentative de forge, elle, ne devra pas provenir de l'oracle. La définition de la résistance à une forge existentielle est donc la suivante. L'attaquant prend en entrée un paramètre de sécurité, des paramètres publics, et une clé publique cible. Il fournit sa tentative de forge $\sigma^\star$ sur un message $m^\star$ de son choix. La probabilité que $\sigma^\star$ soit une signature valide de $m^\star$ doit être négligeable.

Définition 35 (Inforgeabilité) *Soit k un entier, $S = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ un schéma de signatures et $\mathcal{A}$ un adversaire EF-CMA contre S. Considérons les expériences aléatoires suivantes, ou k est un paramètre de sécurité :*

$$\boxed{\textit{Experiment } \mathbf{Exp}^{\mathsf{ef-cma}}_{S,\mathcal{A}}(k)}$$
$$\begin{array}{l} \text{params} \leftarrow S.\mathsf{Setup}(k) \\ (pk, sk) \leftarrow S.\mathsf{KeyGen}(k, \text{params}) \\ (m^\star, \sigma^\star) \leftarrow \mathcal{A}^{\Sigma}(k, \textsf{params}, pk) \\ \textit{Retourner } S.\mathsf{Verify}(k, \text{params}, pk, m^\star, \sigma^\star) \end{array}$$

On définit alors le succès *de l'adversaire $\mathcal{A}$, comme étant*

$$\mathbf{Succ}^{\textit{ef-cma}}_{S,\mathcal{A}}(k) = \Pr\left[\mathbf{Exp}^{\textit{ef-cma}}_{S,\mathcal{A}}(k) = 1\right].$$

Étant donné $\tau \in \mathbb{N}$ et $\varepsilon \in [0,1]$, le schéma S est dit (k, τ, ε)-EF-CMA sûr, si aucun adversaire EF-CMA $\mathcal{A}$ fonctionnant en temps τ, n'a un succès $\mathbf{Succ}^{\textit{ef-cma}}_{S,\mathcal{A}}(k) \geq \varepsilon$.

3.2.2 Diverses signatures utiles par la suite

Parmi les signatures classiques, la plus célèbre est la variante *Full Domain Hash* de la signature RSA (voir figure 2.2. Ce n'est pas le message, mais le haché de celui-ci par une fonction de hachage h, qui est élevé à la puissance d (la clé secrète). C'est la méthode classique dite "hache-et-signe" (*hash-and-sign*[11]). Si la fonction h renvoie des éléments aléatoirement distribués dans $(\mathbb{Z}/N\mathbb{Z})^*$ (d'où le nom de *Full Domain Hash*), Bellare et Rogaway ont montré en 1993 dans [23] que ce schéma de signature, dans le modèle de l'oracle aléatoire, résiste aux forges existentielles sous une attaque à messages choisis adaptative, si le problème RSA est difficile. Coron a amélioré ce résultat en 2001 dans [63] en donnant une meilleure réduction au problème RSA. Bellare et Rogaway ont par la suite proposé dans [25] une construction[12] appelée PSS (*Probabilistic Signature Scheme*), qui offre une réduction fine au problème RSA.

Un travail important a été fait pour obtenir des réductions aussi fines que possibles aux problèmes sous-jacents à la sécurité. La meilleure réduction que l'on peut espérer est la suivante : un adversaire qui casse un schéma avec une probabilité ε et en temps t peut être utilisé pour casser un problème difficile avec une probabilité ε', et en temps t', où $\varepsilon' \approx \varepsilon$ et $t' \approx t$. Coron a montré dans [64] que pour obtenir une sécurité fine dans le modèle hache-et-signe, un message doit avoir plusieurs signatures. Par ailleurs, il montre que la taille de l'aléa nécessaire dans PSS est de 30 bits, et cette taille est optimale (Bellare et Rogaway proposaient un aléa d'une longueur de 180 bits). Dans [105], Goh et Jarecki ont proposé le premier schéma de signatures basé sur le logarithme discret avec une réduction fine à CDH. Katz et Wang ont donné dans [133] une variante de PSS dans laquelle ils utilisent un seul bit d'aléa, chaque message ayant alors deux signatures possibles, tout en gardant une réduction fine (*tight*). Ils améliorent par ailleurs le schéma de Goh et Jarecki.

Parmi les signatures dont la sécurité est prouvée dans le modèle standard, on peut citer par exemple le schéma de Cramer et Shoup [66], qui repose sur la difficulté du problème RSA flexible, comme le schéma de Gennaro, Halevi et Rabin [100]. On peut aussi citer le schéma de Boneh, Mironov et Shoup [40], basé sur les couplages, dont la sécurité repose sur CDH.

[11] dans [177] est proposée une alternative à ce paradigme : les signatures *jumelles*

[12] qui contrairement à FDH ne s'applique pas à toute permutation à sens-unique à trappe, puisqu'il faut utiliser le fait que RSA est *self random reducible*

3.2.2.1 Signatures courtes

Outre la construction du premier schéma de chiffrement basé sur l'identité efficace par Boneh et Franklin [35, 36], l'introduction des couplages en cryptographie a permis à Boneh, Lynn et Shacham de proposer un élégant et très simple schéma de signatures courtes à Asiacrypt'01 [38], paru ensuite dans le *Journal of Cryptology* [39].

L'élégance et la simplicité découlent naturellement de l'existence de groupes dans lesquels le problème Diffie-Hellman décisionnel est facile grâce aux couplages [131]. Les signatures BLS sont décrites dans la figure 3.2. La petite taille de ces signatures provient du fait que des couplages existent sur le groupe des points de courbes elliptiques, et qu'on obtient alors des signatures qui sont essentiellement représentées par un point sur cette courbe et dont la taille vaut environ 160 bits. Dans leur article, Boneh *et al.* prouvent la sécurité de leur schéma, dans le modèle de l'oracle aléatoire, en montrant qu'il est équivalent au problème CDH. Comme nous le verrons plus tard, ce schéma est un plongement du schéma de signatures indéniables de Chaum et van Antwerpen [57] dans un groupe possédant un couplage. La réduction de l'inforgeabilité des signatures BLS est donnée par le théorème suivant :

Théorème 10 *Soit k un entier, BDHGen un générateur de groupes BDH symétrique et BLS le schéma de signatures associé. Dans le modèle de l'oracle aléatoire, pour tout adversaire EF-CMA $\mathcal{A}$ contre BLS, de paramètre de sécurité k, ayant un succès $\mathbf{Succ}^{\mathsf{ef-cma}}_{\mathsf{BLS},\mathcal{A}}(k) = \varepsilon$ en temps τ, et pouvant faire $q_{\mathcal{H}}$ requêtes à l'oracle aléatoire et q_Σ requêtes à l'oracle de signature, il existe un adversaire $\mathcal{D}$ contre CDH, de succès $\mathbf{Succ}^{\mathsf{cdh}}_{\mathsf{BDHGen},D}(k) = \varepsilon'$ en temps τ' tel que*

$$\begin{cases} \varepsilon' \geq \dfrac{\varepsilon}{\exp(1)(q_\Sigma + 1)} \\ \tau' \leq \tau + c_{\mathbb{G}}(q_{\mathcal{H}} + 2q_\Sigma). \end{cases}$$

où $c_{\mathbb{G}}$ est une constante dépendante de $\mathbb{G}$.

Les techniques de Goh et Jarecki [105], ou de Katz et Wang peuvent être utilisées pour rendre fine la réduction à CDH.

Boneh *et al.* donnent aussi dans [38] une construction de fonctions de hachage qui permet de hacher des messages sur une courbe elliptique. Cette technique est assez coûteuse (mais a fait l'objet d'améliorations, en particulier par Barreto et Kym [10]), et s'ajoute au calcul de deux couplages dans la vérification. Cela a motivé la naissance d'une variante de ce protocole par Zhang, Safavi-Naini et Susilo à PKC'04 [248] et par Boneh et Boyen à Eurocrypt'04 [34], dans laquelle il n'est pas nécessaire de hacher sur la courbe, et qui nécessite un couplage de moins à calculer lors de la vérification. Zhang *et al.* montrent que leur schéma est sûr dans le modèle de l'oracle aléatoire sous une hypothèse plus forte que pour BLS, à savoir l'hypothèse k-CAA (voir la définition 26). Boneh et Boyen, eux, montrent que leur variante est sûre dans le modèle standard, sous l'hypothèse q-SDH (voir la définition 23). Cette variante, qui servira à construire certains des schémas présentés dans cette thèse, est décrite dans la figure 3.3.

La deuxième clé publique dans l'élément $m + yr$, permet d'insérer de l'information liée à q-SDH dans l'aléa r et donc de démontrer le théorème suivant sans oracle aléatoire :

Théorème 11 *Soit k un entier, BDHGen un générateur de groupes BDH et BB le schéma de signatures associé. Pour tout adversaire $\mathsf{EF\text{-}CMA}$ $\mathcal{A}$ contre BB, de paramètre de sécurité k, ayant un succès $\mathbf{Succ}^{\mathsf{ef-cma}}_{\mathsf{BLS},\mathcal{A}}(k) = \varepsilon$ en temps τ, et pouvant faire q_Σ requêtes à l'oracle de signature, il*

Génération des paramètres publics : $\mathsf{BLS.Setup}$

• k est un paramètre de sécurité,
• $(q, P, \mathbb{G}, \mathbb{H}, e)$ est généré par $\mathsf{BDHGen}(k)$
• Soit $[\{0,1\}^* \longrightarrow \mathbb{G}]$ une famille de fonctions de hachage et H un élément pris au hasard dans cette famille.

Génération des clés : $\mathsf{BLS.KeyGen}$

• $a \in [\![1, q-1]\!]$ est la clé secrète, la clé publique est $P_A = aP$

Signature : $\mathsf{BLS.Sign}$

• Étant donné un message m, Alice calcule le point $\sigma = aH(m)$

Vérification : $\mathsf{BLS.Verify}$

• Bob vérifie que $(P_A, H(m), \sigma)$ est un triplet Diffie-Hellman valide, soit :
$e(P_A, H(m)) = e(\sigma, P)$

FIG. 3.2 – Signatures BLS

existe un adversaire $\mathcal{D}$ contre q_Σ-SDH, de succès $\mathbf{Succ}^{q_\Sigma\text{-sdh}}_{\mathsf{BDHGen},D}(k) = \varepsilon'$ *en temps* τ' *tel que*

$$\begin{cases} \varepsilon' \geq \dfrac{\varepsilon}{2} - \dfrac{q_\Sigma}{2^k} \\ \tau' \leq \tau + O(q_\Sigma^2 T_{Exp}). \end{cases}$$

3.2.2.2 Signatures d'anneau

Les signatures d'anneau (*ring signatures*) ont été introduites par Camenisch dans [47], puis plus précisément par Rivest, Shamir et Tauman dans [209]. Elles tiennent un rôle important parmi les signatures utilisées dans un contexte de cryptographie de groupe, où l'anonymat est primordial, au même titre que, par exemple, les signatures de groupes [51, 58]. Nous donnons ici leur définition, puisqu'elles nous serviront de base à la construction de schémas de signatures à vérificateurs désignés.

Informellement, on considère un groupe de signataires potentiels, auxquels est associée une clé publique. Au sein de ce groupe, un membre particulier va réellement produire une signature (qui contiendra la totalité des clés publiques du groupe, signataire compris). Ces signatures ont de spécial le fait qu'un vérificateur sera incapable de déterminer l'identité du vrai signataire parmi n signataires potentiels avec une probabilité meilleure que $1/n$, même en connaissant plusieurs signatures de ce signataire. Cette propriété porte le nom d'*ambiguïté du signataire*, et peut être obtenue de manière calculatoire ou inconditionnelle.

Définition 36 (Signature d'anneau) *Étant donné deux entiers n (le nombre de membres de l'anneau) et k, un* schéma de signatures d'anneau *Ring de paramètre de sécurité k est composé de 4 algorithmes :*

Génération des paramètres publics : BB.Setup

• k est un paramètre de sécurité,
• BDHGen un générateur de groupes BDH et $(q, P_0, \mathbb{G}_0, P_1, \mathbb{G}_1, \mathbb{H}, e)$ est généré par BDHGen(k)

Génération des clés : BB.KeyGen

• $(x, y) \in [\![1, q-1]\!]^2$ est la clé secrète et la clé publique est $(P_0', P_1', P_A^x, P_A^y, z)$ où P_0' et P_1' sont des générateurs de $\mathbb{G}_0$ et $\mathbb{G}_1$, $P_A^x = xP_1'$, $P_A^y = yP_1'$ et $z = e(P_0', P_1')$.

Signature : BB.Sign

• Étant donné un message m, Alice tire au hasard $r \in [\![1, q-1]\!]$
• Elle calcule $\sigma = \dfrac{1}{x+m+ry} P_0'$ et la signature est (σ, r)

Vérification : BB.Verify

• Étant donné m, la clé publique d'Alice et la signature (σ, r), Bob vérifie que

$$e(\sigma, P_A^x + mP_1' + rP_A^y) = e(P_0', P_1')$$

Fig. 3.3 – Signatures Boneh-Boyen

– génération des paramètres publics *Ring.Setup : algorithme probabiliste prenant en entrée un entier k et renvoyant les* paramètres publics *;*
– génération des clés *Ring.SKeyGen : algorithme probabiliste prenant en entrée un entier k, les paramètres publics et produisant une paire de clés (pk, sk) ;*
– signature *Ring.Sign : algorithme probabiliste prenant en entrée un entier k, les paramètres publics, un message m, les clés publiques des n utilisateurs et une clé secrète d'un membre spécifique. Sa sortie σ est une* signature d'anneau *du message m. Elle contient les n clés publiques des membres de l'anneau.*
– vérification *Ring.Verify : algorithme déterministe prenant en entrée un entier k, les paramètres publics, un message m et une chaîne de bits σ et testant si σ est une signature d'anneau valide du message m.*

Un schéma de signature d'anneau doit vérifier les propriétés suivantes :

– *consistance,*
– *inforgeabilité,*
– *ambiguïté du signataire (*source hiding *ou* signer's ambiguity*).*

Le schéma de Rivest *et al.* est basé sur RSA et sur des *fonctions de combinaisons*. Il a été amélioré dans [45] par Bresson, Stern et Szydlo, et modifié pour obtenir une version à seuil. Les signatures d'anneau qui nous serviront par la suite, sont celles basées sur le logarithme discret. On peut citer par exemple le schéma de Abe, Ohkubo et Suzuki [1], celui de Herranz et Sáez [122], et le schéma de Boneh, Gentry, Lynn et Shacham basé sur les couplages. Nous détaillons ce dernier dans la figure 3.4. Sa sécurité, dans le modèle de l'oracle aléatoire, repose sur la difficulté de CDH.

Génération des paramètres publics : $\mathsf{BGLS.Setup}$

• k est un paramètre de sécurité,
• BDHGen un générateur de groupes BDH symétrique et $(q, P, \mathbb{G}, \mathbb{H}, e)$ est généré par $\mathsf{BDHGen}(k)$
• Soit $[\{0,1\}^* \longrightarrow \mathbb{G}]$ une famille de fonctions de hachage H un élément pris au hasard dans cette famille.

Génération des clés : $\mathsf{BGLS.KeyGen}$

• $x_i \in [\![1, q-1]\!]$ est la clé secrète et la clé publique est P_i où $P_i = x_i P$.

Signature : $\mathsf{BGLS.Sign}$

• Étant donné un message m, et les clés publiques $P_1,\ldots,P_n$ des membres du groupe, le signataire s $(1 \leq s \leq n)$ tire au hasard $r_i \in [\![1, q-1]\!]$ pour $i \in [\![1, n]\!] \setminus \{s\}$.
• Il calcule $\sigma_s = x_s^{-1}\Big(H(m) - \sum_{\substack{i=1 \\ i \neq s}}^{n} r_i P_i\Big)$ et pose $\sigma_i = r_i P$ pour $i \in [\![1, n]\!] \setminus \{s\}$. La signature est $\sigma = (\sigma_1, \ldots, \sigma_n)$

Vérification : $\mathsf{BGLS.Verify}$

• Étant donné (m, σ), les clés publiques $P_1,\ldots,P_n$ des membres du groupe, un vérificateur teste que

$$e(H(m), P) = \prod_{i=1}^{n} e(\sigma_i, P_i)$$

FIG. 3.4 – Signatures d'anneau de Boneh, Gentry, Lynn et Shacham

Les signatures d'anneau ont l'avantage (ou l'inconvénient...), contrairement aux signatures de groupes, d'être naturellement dynamiques, en ce sens que le signataire choisit au moment de la signature les membres de l'"anneau". Un des gros désavantages de ces signatures provient de la taille qui croît de façon linéaire avec le nombre de membres dans l'anneau. Il existe des techniques pour obtenir une taille de signature constante (voir en particulier l'article de Dodis, Kiayias, Nicolosi et Shoup [78]).

3.3 Remarques

3.3.1 Enregistrement des clés des utilisateurs

Dans un contexte multi-acteurs, il est important que les utilisateurs enregistrent leur clés publiques auprès d'une autorité de certification. C'est le principe des infrastructures à clés publiques (*public-key infrasctructure* - PKI). Aussi, dans les schémas de signatures que nous présenterons par la suite, il sera sous-entendu que les clés des différents utilisateurs auront été préalablement enregistrées auprès d'une autorité. La procédure suivie inclura, par exemple, une preuve de connaissance de logarithme discret de la part de l'utilisateur. L'autorité produisant ensuite son certificat.

Dans les modèles de sécurité que nous considérerons, lorsqu'un attaquant souhaitera obtenir des signatures dont la génération suppose l'utilisation d'une clé publique d'un utilisateur tiers, il pourra choisir librement ce dernier. L'enregistrement systématique imposera que l'attaquant connaisse la clé secrète correspondant à la clé publique utilisée dans la signature. Par ailleurs, ceci nous permettra d'extraire ce secret détenu par l'attaquant en rembobinant le processus d'enregistrement de la clé publique, et donc de simuler aisément les algorithmes utilisant le secret choisi par l'attaquant. Si la preuve de connaissance d'un logarithme discret est celle de Schnorr (figure 1.3) par exemple la simulation procède de la façon suivante pour extraire le secret : au premier enregistrement d'une clé x, l'attaquant tire un r aléatoirement, puis envoie g^r à l'autorité. Celle-ci envoie alors une question e à l'attaquant qui retourne $d = r + ex$. À ce moment là, la simulation est rembobinée par l'algorithme de réduction qui la relance en utilisant les mêmes bandes aléatoires pour l'attaquant, qui procède donc une nouvelle fois à l'enregistrement des clés en commençant de la *même façon* que la première fois. Il envoie donc g^r pour le même r que dans le premier enregistrement, mais l'algorithme de réduction, lui envoie un $c' \neq c$ à l'attaquant qui envoie alors $d' = r + c'x$. De la connaissance de d et d', l'algorithme de réduction extrait $x = (y - y')/(c - c')$.

3.3.2 Preuves de sécurité

Parmi les preuves de sécurité présentées dans le chapitre suivant, quelques unes sont présentées sous forme d'une succession de jeux. Cette façon d'écrire les preuves a été introduite par Kilian et Rogaway en 1996 [136] et popularisée par Shoup, en particulier dans [228]. L'idée est de considérer dans un premier temps l'attaque telle qu'elle se déroule lorsque l'attaquant est confronté à son challenger. C'est le jeu "0", l'attaque réelle. L'attaquant et son challenger sont des machines de Turing probabilistes, et définissent un espace de probabilité. Le reste de la preuve est alors constitué d'un certain nombre de jeux Jeu_1, Jeu_2, ...,Jeu_n. On part d'un événement S (par exemple, "l'attaque réussit"), puis, pour chaque jeu Jeu_i, où $i \in [\![1, n]\!]$, on définit un événement S_i, lié à S. Lors de la preuve, on montre que $\Pr[S_i]$ est proche de $\Pr[S_{i+1}]$ à un terme négligeable près pour $i \in [\![0, n-1]\!]$, et finalement $\Pr[S_n]$ est essentiellement la probabilité de l'événement cible (par exemple "l'algorithme a cassé un problème difficile"). La conclusion de la preuve consiste à estimer à quel point $\Pr[S_n]$ est proche de $\Pr[S]$. Shoup note trois types de transitions entre les jeux :

- des transitions basées sur l'indistinguabilité : un changement anodin est fait dans un jeu, qui, s'il est détecté par l'adversaire, permettrait de construire un distingueur de deux distributions qui ne sont pas distinguables. On conclut alors que $|\Pr[S_i] - \Pr[S_{i+1}]|$ est négligeable ;

- des transitions liées à l'apparition d'un événement d'échec F. Les jeux i et $i+1$ sont les mêmes, à moins que cet événement ait lieu, ce qui est équivalent (si les distributions de probabilité sont les mêmes) à :

$$S_i \wedge \neg F \Longleftrightarrow S_{i+1} \wedge \neg F.$$

L'utilisation du lemme suivant permet de conclure que si $\Pr[F]$ est négligeable, $\Pr[S_i]$ est suffisamment proche de $\Pr[S_{i+1}]$. La notoriété de ce lemme évident lui a apporté le nom de *lemme de différence*[13].

Lemme 1 (Lemme de différence) *Soit A, B et F des événements définis dans un certain espace de probabilité, et on suppose que $A \wedge \neg F \Longleftrightarrow B \wedge \neg F$. Alors*

$$|\Pr[A] - \Pr[B]| \leq \Pr[F].$$

Preuve :

La preuve est un calcul élémentaire de probabilité :

$$\begin{array}{rcl} |\Pr[A] - \Pr[B]| & = & |\Pr[A \wedge F] + \Pr[A \wedge \neg F] - \Pr[B \wedge F] - \Pr[B \wedge \neg F]| \\ & = & |\Pr[A \wedge F] - \Pr[B \wedge F]| \text{ car } A \wedge \neg F \Longleftrightarrow B \wedge \neg F \\ & \leq & \Pr[F] \text{ puisque } 0 \leq \Pr[X \wedge F] \leq \Pr[F],\ X \in \{A, B\}; \end{array}$$

□

- des transitions conceptuelles, qui ne sont que des façons de réécrire certaines quantités, et qui préparent les deux transitions précédentes : on a alors $\Pr[S_i] = \Pr[S_{i+1}]$.

[13] quand ce n'est pas *lemme de Shoup*

Deuxième partie

Signatures électroniques : variantes

1

Préambule

1.1 Introduction

On demande aux signatures numériques beaucoup plus, en fait, qu'aux signatures manuscrites. Et pour cause : leur potentielle dissémination, liée à l'évolution des technologies de communication, nécessite une maîtrise parfaite de leur sécurité. Devant le nombre des applications dans lesquelles sont indispensables les notions d'authentification, d'intégrité, d'anonymat, il s'avère que les signatures numériques ordinaires ne sont pas toujours les outils adequats. De fait, de nombreuses variantes ont été inventées pour répondre à des problèmes pratiques, liés au vote électronique, aux transactions financières (monnaie électronique, enchères,...) ou aux signatures de contrat par exemple.

Nous allons dans la suite développer certaines de ces signatures "spéciales" en essayant d'apporter à nos schémas des propriétés d'anonymat qui enrichissent leur utilisation mais aussi des améliorations en terme de performances. Quand on parle d'efficacité, bien sûr, on peut se placer à plusieurs niveaux. Il peut s'agir du coût (temps de calcul) de la signature, ou de celui de la vérification. Il peut aussi s'agir de la taille de la signature. L'utilisation des couplages déséquilibre en général le coût de la génération et de la vérification. En effet, dans la grande majorité des cas, le calcul d'une signature se fait au moyen de quelques exponentiations sur une courbe. La vérification, quant à elle, nécessite le calcul - coûteux - d'un couplage. L'avantage d'utiliser les courbes munies d'une application bilinéaire admissible reste que la signature devient en général un point sur une courbe (et un aléa), ce qui la rend relativement courte. Or de nombreuses applications ont rendu pertinente l'étude et la construction de schémas de signatures courtes. Leur utilisation dans des environnements où la bande passante est faible est tout à fait appropriée, ou bien lorsque la signature doit par exemple être lue puis recopiée par un utilisateur humain[14], lorsque la signature doit tenir dans des paquets de taille fixée, lorsqu'elle doit être contenue dans une carte à puce, sur un timbre (voir [178]).

Bien sûr toutes les signatures, pas seulement les signatures classiques, tirent un avantage de leur taille. Ainsi, toutes les signatures présentées dans la suite sont des signatures dites "courtes" et pour lesquelles (au moins) la génération de la signature a un faible coût.

L'histoire des signatures du chapitre suivant commence en 1989 avec l'article fondateur du concept de signatures indéniables écrit par Chaum et van Antwerpen [57]. Après plus de 10 ans, et quelques mésaventures, Okamoto et Pointcheval [188] prouvent la sécurité de la version *Full Domain Hash* de ces signatures indéniables. À cette occasion, les auteurs introduisent le concept de problèmes *gap*, ces problèmes calculatoires que l'on essaye de résoudre avec l'aide

[14]pas par une machine de Turing

d'un oracle pour le problème décisionnel associé. L'apparition des couplages de Weil et Tate en cryptographie [91, 165] a matérialisé ces oracles décisionnels, et a donné naissance à des groupes dans lesquels DDH est facile alors que CDH est difficile [131]. En 2001, Boneh, Lynn et Shacham ont exploité ces groupes *gap* en y plongeant la signature indéniable de Chaum et van Antwerpen. Ce faisant, ils l'ont transformée en une signature classique, universellement vérifiable, en deux calculs de couplage. Ces signatures ont été améliorées par Zhang, Safavi-Naini et Susilo [248] et par Boneh et Boyen [34]. Ces trois signatures constituent le cœur des signatures que nous allons présenter. Elles sont assemblées, modifiées, déplacées pour obtenir des signatures courtes aux propriétés souhaitées.

1.2 L'astuce xyz

Les protocoles de signatures "spéciales" proposés par la suite sont construits grâce à des applications bilinéaires telles que celles décrites dans la partie 2.3. L'avantage apporté par ces objets, outre la bilinéarité, vient du fait qu'en général la signature (ou une partie de celle-ci) consiste en des points de la courbe sous-jacente, et donc la taille de ces signatures est petite.

La simple observation suivante nous a permis d'apporter aisément des propriétés en général difficiles à obtenir dans des schémas de signatures.

1.2.1 Le problème xyz-DDH

Dans le contexte de groupes munis de couplages, la version décisionnelle du problème Diffie-Hellman est facile. Une extension naturelle de ce problème est la suivante :

Problème xyz-Diffie-Hellman décisionnel [xyz-DDH] : soit $(\mathbb{G}, +)$ un groupe d'ordre un grand premier, noté q, engendré par P. Soit $(x, y, z) \in [\![1, q-1]\!]^3$. Étant donné $(P, xP, yP, zP, Q) \in \mathbb{G}^5$, décider si $Q = xyzP$.

Il est effectivement similaire au problème classique DDH. Cependant, contrairement à ce dernier, xyz-DDH semble difficile même en présence de couplage. Sa version calculatoire est équivalente au problème CDH classique, et il est plus simple que le problème DBDH. Après avoir donné une définition quantitative de l'hypothèse associée, nous montrerons que ce problème est difficile dans un groupe générique.

Définition 37 (xyz-DDH) *Soit k un entier et BDHGen un générateur de groupes BDH symétrique. Soit D un attaquant qui prend en entrée $(q, P, \mathbb{G}, \mathbb{H}, e)$ un quintuplet généré par $\mathsf{BDHGen}(k)$ et $(X, Y, Z, Q) \in \mathbb{G}^4$ et qui renvoie un bit. On considère les expériences aléatoires suivantes, où k est un paramètre de sécurité et $r \in \{0, 1\}$:*

$\boxed{\textit{Experiment } \mathbf{Exp}^{\mathsf{xyz-ddh-r}}_{\mathsf{BDHGen},D}(k)}$
params $= (q, P, \mathbb{G}, \mathbb{H}, e) \leftarrow \mathsf{BDHGen}(k)$
$(x, y, z) \xleftarrow{R} [\![1, q-1]\!]^3$
$X \leftarrow xP, Y \leftarrow yP, Z \leftarrow zP$
Si $r = 0$ *alors* $Q \leftarrow xyzP$ *sinon* $Q \xleftarrow{R} \mathbb{G}$
$d \leftarrow D(\text{params}, X, Y, Z, Q)$
Retourner d

*On définit alors l'*avantage *de D pour résoudre le problème xyz-DDH comme étant :*

$$\mathbf{Adv}^{\textsf{xyz-ddh}}_{\textsf{BDHGen},D}(k) = \left|\Pr\left[\mathbf{Exp}^{\textsf{xyz-ddh-0}}_{\textsf{BDHGen},D}(k) = 1\right] - \Pr\left[\mathbf{Exp}^{\textsf{xyz-ddh-1}}_{\textsf{BDHGen},D}(k) = 1\right]\right|.$$

Étant donné $\tau \in \mathbb{N}$ et $\varepsilon \in [0,1]$, le problème xyz-DDH est dit (k, τ, ε)-difficile si aucun adversaire D, fonctionnant en temps τ, n'a un avantage $\mathbf{Adv}^{\textsf{xyz-ddh}}_{\textsf{BDHGen},D}(k) \geq \varepsilon$.

Nous rappelons la définition d'une fonction d'encodage et d'un algorithme générique, puis nous donnons une idée de la difficulté générique de xyz-DDH.

Définition 38 (Shoup) *Soit p un nombre premier.*

- *Une* fonction d'encodage *sur le groupe (additif) $(\mathbb{Z}/p\mathbb{Z}, +)$ est une application injective $\sigma : \mathbb{Z}/p\mathbb{Z} \longrightarrow \{0,1\}^n$ pour un certain entier $n > 0$.*

- *Un* algorithme générique *$\mathcal{A}$ pour $(\mathbb{Z}/p\mathbb{Z}, +)$ est un algorithme probabiliste prenant en entrée une* liste d'encodages *L initialisée à $(\sigma(x_1), \ldots, \sigma(x_\ell))$, pour un $\ell \in \mathbb{N}$ et où σ est une fonction d'encodage et $x_i \in \mathbb{Z}/p\mathbb{Z}$ pour $i \in [\![1, \#L]\!]$. Durant son exécution, l'algorithme peut interroger un oracle en donnant deux indices i et j dans $[\![1, \#L]\!]$, et un bit de signe. L'oracle retourne alors $\sigma(x_i \pm x_j)$ suivant le signe, ce nouvel encodage étant rajouté à la liste d'encodages. Finalement l'algorithme termine et sa sortie est notée $\mathcal{A}(\sigma(x_1), \ldots, \sigma(x_\ell))$.*

Nous nous plaçons donc maintenant dans des groupes $\mathbb{G}$ (engendré par P) et $\mathbb{H}$ (engendré par $e(P,P)$) d'ordre q, reliés par une application bilinéaire admissible $e : \mathbb{G} \times \mathbb{G} \longrightarrow \mathbb{H}$. Les éléments du groupe sont donc représentés par une unique suite de bits aléatoires. La seule opération que peut effectuer l'adversaire sur ces suites est le test d'égalité. Il fait appel, pour les opérations de groupes, ou les calculs de couplage, à des oracles.

Théorème 12 *Soit un entier premier q, et $S \subset \{0,1\}^*$ un ensemble d'au moins q éléments. Soit $\mathcal{A}$ un algorithme générique pour $(\mathbb{Z}/q\mathbb{Z}, +)$ qui résout le problème xyz-DDH faisant au plus m requêtes à l'oracle de calcul dans le groupe $\mathbb{G}$, à l'oracle de calcul dans le groupe $\mathbb{H}$ et à l'oracle de calcul de couplage. Soit $(a,b,c,d) \in (\mathbb{Z}/q\mathbb{Z})^4$ tirés au hasard, $\sigma_{\mathbb{G}} : \mathbb{Z}/q\mathbb{Z} \longrightarrow S$ et $\sigma_{\mathbb{H}} : \mathbb{Z}/q\mathbb{Z} \longrightarrow S$ deux fonctions d'encodage, et s un bit aléatoire. On pose $\omega_0 = abc$ et $\omega_1 = d$. Alors*

$$\Pr\left[\mathcal{A}(\sigma(1), \sigma(a), \sigma(b), \sigma(c), \sigma(\omega_s), \sigma(\omega_{1-s})) = s\right] - \frac{1}{2} \leq 6\frac{m^2}{q},$$

où la probabilité est prise sur les tirages aléatoires de a, b, c, d dans $\mathbb{Z}/q\mathbb{Z}$, des fonctions d'encodage $\sigma_{\mathbb{G}}$ et $\sigma_{\mathbb{H}}$ et des bits d'aléa utilisés par $\mathcal{A}$.

Preuve :

On considère un algorithme $\mathcal{B}$ qui joue le jeu suivant avec $\mathcal{A}$.

$\mathcal{B}$ tire au hasard un quadruplet $(a,b,c,d) \in [\![1, q-1]\!]^4$ et maintient une liste $\sigma_{\mathbb{G}}$-list qu'il initialise à $\{(\xi, 1), (\xi_a, A), (\xi_b, B), (\xi_c, C), (\xi_t, T), (\xi_u, U)\}$, où les ξ_j sont des suites de bits distinctes, telles que $\xi_j = \sigma_{\mathbb{G}}(j)$ est la représentation de jP et $(t,u) = (abc, d)$ ou (d, abc). Le dernier terme du triplet est un polynôme de $\mathbb{Z}/q\mathbb{Z}[A,B,C,T,U]$. Il fait de même avec une $\sigma_{\mathbb{H}}$-list : il l'initialise à $\{(\xi', 1), (\xi'_a, A), (\xi'_b, B), (\xi'_c, C), (\xi'_t, T), (\xi'_u, U)\}$. $\mathcal{B}$ fournit à $\mathcal{A}$ les deux listes $(\xi, \xi_a, \xi_b, \xi_c, \xi_u, \xi_t)$ et $(\xi', \xi'_a, \xi'_b, \xi'_c, \xi'_u, \xi'_t)$.

À chaque interaction avec l'oracle de calcul dans $\mathbb{G}$, $\mathcal{A}$ apprend l'encodage d'un x_iP, où $x_i \in \mathbb{Z}/q\mathbb{Z}$ est tel que $x_i = L_i(a,b,c,t,u)$, où L_i est un polynôme linéaire appartenant à $\mathbb{Z}/q\mathbb{Z}[A,B,C,T,U]$ que $\mathcal{B}$ peut déduire des questions précédentes à l'oracle. Il en est de même avec l'oracle de calcul dans $\mathbb{H}$: $\mathcal{A}$ obient l'encodage d'un élément $e(P,P)^{x_i}$. De l'oracle de calcul de couplage, $\mathcal{A}$ apprend l'encodage d'un élément $e(x_iP, x_jP)$ (où $(x_i,x_j) \in (\mathbb{Z}/q\mathbb{Z})^2$) qui correspond à l'encodage dans $\mathbb{H}$ de $e(P,P)^{x_ix_j}$, et $x_ix_j = L_i(a,b,c,t,u)L_j(a,b,c,t,u)$ qui est un polynôme de degré inférieur ou égal à 2.

Simulation des lois de groupes : lorsque $\mathcal{A}$ s'adresse à un oracle de calcul dans un groupe avec un couple du type $((i,j),\pm)$ où $(i,j) \in [\![1,\#\sigma_{\mathbb{X}}\text{-list}]\!]^2$, $(\mathbb{X} \in \{\mathbb{G},\mathbb{H}\})$, $\mathcal{B}$ calcule le polynôme $L = L_i \pm L_j \in \mathbb{Z}/q\mathbb{Z}[A,B,C,T,U]$. S'il existe $k \in [\![1,\#\sigma_{\mathbb{X}}\text{-list}]\!]$ tel que $L = L_k$, alors $\mathcal{B}$ répond ξ_k, sinon, il choisit ξ aléatoirement dans $S^* \setminus \{\xi_i\}_{i \leq \#\sigma_{\mathbb{X}}\text{-list}}$, le renvoie à $\mathcal{A}$, et met à jour sa $\sigma_{\mathbb{X}}$-list avec ξ et L.

Simulation de l'oracle de calcul de couplage : lorsque $\mathcal{A}$ s'adresse à l'oracle de calcul d'un couplage avec $((i,j))$ où $(i,j) \in [\![1,\#\sigma_{\mathbb{G}}\text{-list}]\!]^2$, $\mathcal{B}$ calcule le polynôme $L = L_iL_j$. S'il existe $k \in [\![1,\#\sigma_{\mathbb{H}}\text{-list}]\!]$ tel que $L' = L'_k$, alors $\mathcal{B}$ répond ξ'_k, sinon, il choisit ξ' aléatoirement dans $S \setminus \{\xi'_i\}_{i \leq \#\sigma_{\mathbb{H}}\text{-list}}$, le renvoie à $\mathcal{A}$, et met à jour sa $\sigma_{\mathbb{H}}$-list avec ξ' et L'.

À la fin du jeu, $\mathcal{A}$ renvoie un bit $s_{\mathcal{A}}$, à la suite de m requêtes à $\mathcal{B}$.

Notons maintenant $\mathcal{E}_{\mathcal{S}}$ l'ensemble

$$\{(a,b,c,d) \in (\mathbb{Z}/q\mathbb{Z})^4,\ \exists (i,j) \in [\![1,\#\sigma_{\mathbb{G}}\text{-list}]\!]^2 : L_i(a,b,c,t,u) = L_j(a,b,c,t,u) \text{ et } L_i \neq L_j\}$$

$$\bigcup$$

$$\{(a,b,c,d) \in (\mathbb{Z}/q\mathbb{Z})^4,\ \exists (i,j) \in [\![1,\#\sigma_{\mathbb{H}}\text{-list}]\!]^2 : L'_i(a,b,c,t,u) = L'_j(a,b,c,t,u) \text{ et } L'_i \neq L'_j\}.$$

Si, pour tout $(i,j) \in [\![1,\#\sigma_{\mathbb{X}}\text{-list}]\!]^2$ $(\mathbb{X} \in \{\mathbb{G},\mathbb{H}\})$ tel que $L_i \neq L_j$, $\xi_i \neq \xi_j$, cela signifie que $\mathcal{A}$ a appris des encodages de valeurs distinctes. Puisque les ξ_l sont tirés aléatoirement, ils ne fournissent aucune information à $\mathcal{A}$. Ainsi, la seule façon pour $\mathcal{A}$ d'obtenir de l'information, c'est que (a,b,c,d) appartienne à $\mathcal{E}_{\mathcal{S}}$.

Nous allons tout d'abord montrer que si $(a,b,c,d) \in (\mathbb{Z}/q\mathbb{Z})^4 \setminus \mathcal{E}_{\mathcal{S}}$, alors

$$\Pr[\mathcal{A}(\sigma(1),\sigma(a),\sigma(b),\sigma(c),\omega_s,\omega_{1-s}) = s] = 1/2.$$

Pour cela, nous allons montrer que $\mathcal{B}$ peut toujours convaincre $\mathcal{A}$ qu'il s'est trompé en exhibant une fonction d'encodage σ, cohérente avec ses réponses à $\mathcal{A}$ lorsque $s = 1 - s_{\mathcal{A}}$. Nous rappelons que $\mathcal{A}$ répond 0 si $\xi_t = \sigma(abc)$ et 0 sinon.

Considérons le cas où $s_{\mathcal{A}} = 0$: dans sa liste d'initialisations, $\mathcal{B}$ choisit donc $\xi_t = \sigma(d)$ et $\xi_u = \sigma(abc)$, et finit de construire σ en posant $\sigma(L_i(a,b,c,d,abc)) = \xi_i$ pour tout $i \in [\![1,\#\sigma_{\mathbb{G}}\text{-list}]\!]^2$, $\mathcal{B}$ renvoie des valeurs aléatoires dans S distinctes des précédentes. Cette fonction est bien une fonction d'encodage : c'est une fonction si $L_i(a,b,c,d,abc) = L_j(a,b,c,d,abc) \Longrightarrow \xi_i = \xi_j$, or $L_i(a,b,c,d,abc) = L_j(a,b,c,d,abc) \Longrightarrow L_i = L_j$ car $(a,b,c,d) \in (\mathbb{Z}/q\mathbb{Z})^4 \setminus \mathcal{E}_{\mathcal{S}}$. Ainsi, s étant tiré de façon aléatoire par $\mathcal{B}$ après que $\mathcal{A}$ ait envoyé son résultat,

$$\Pr[\mathcal{A}(\sigma(1),\sigma(a),\sigma(b),\sigma(c),\sigma(\omega_s),\sigma(\omega_{1-s})) = s] = 1/2.$$

Il faut maintenant considérer la probabilité ε que, étant donné au plus m distincts polynômes linéaires et des éléments aléatoires $(a,b,c,d) \in (\mathbb{Z}/q\mathbb{Z})^4$, il existe $(i,j) \in [\![1,m]\!]^2$, $i \neq j$ tels que $L_i(a,b,c,d,abc) = L_j(a,b,c,d,abc)$ ou que, étant donné au plus m distincts polynômes de degré au plus 2 et des éléments aléatoires $(a,b,c,d) \in (\mathbb{Z}/q\mathbb{Z})^4$, il existe $(i,j) \in [\![1,m]\!]^2$, $i \neq j$ tels que $L_i(a,b,c,d,abc) = L_j(a,b,c,d,abc)$.

On considère donc le nombre de solutions de l'équation $L_i(x,y,z,t,xyz) = L_j(w,y,z,t,xyz)$, qui est au plus celui de $P(x,y,z,t) = L_i(x,y,z,t,xyz) - L_j(w,y,z,t,xyz) = 0$. Le polynôme P est de degré 3 si les L_i et L_j sont linéaires et de degré 6 si les L_i et L_j sont de degrés 2. Le lemme suivant nous permet de majorer la probabilité qu'un quadruplet aléatoire $(x,y,z,t) \in (\mathbb{Z}/q\mathbb{Z})^4$ annule un tel polynôme P.

Lemme 2 (Shoup - [227]) *Soit p un premier et t un entier non nul.*
Soit $P(X_1,\ldots,X_k) \in \mathbb{Z}/p^t\mathbb{Z}[X_1,\ldots,X_k]$ un polynôme non nul de degré d. Alors, étant donné des éléments aléatoires $x_1,\ldots,x_k \in \mathbb{Z}/p^t\mathbb{Z}$, la probabilité que $P(x_1,\ldots,x_k) = 0$ est au plus d/p.

Puisqu'il y a $\binom{m}{2}$ paires de polynômes (L_i, L_j) à considérer, la probabilité qu'un quadruplet aléatoire $(x,y,z,t) \in (\mathbb{Z}/q\mathbb{Z})^4$ annule P est majorée par :

$$\varepsilon \leq 2\binom{m}{2}\frac{3}{q} + \binom{m}{2}\frac{6}{q} \leq 6\frac{m^2}{q}.$$

En utilisant la majoration grossière

$$\Pr\left[\mathcal{A}(\sigma(1),\sigma(a),\sigma(b),\sigma(c),\sigma(\omega_s),\sigma(\omega_{1-s})) = s\right] \leq 1$$

lorsque $(a,b,c,d) \in \mathcal{E}_{\mathcal{S}}$, on obtient finalement

$$\Pr\left[\mathcal{A}(\sigma(1),\sigma(a),\sigma(b),\sigma(c),\sigma(\omega_s),\sigma(\omega_{1-s})) = s\right] - \frac{1}{2} \leq 6\frac{m^2}{q}.$$

□

Ce problème va nous permettre d'obtenir des compromis entre authenticité et anonymat grâce aux remarques suivantes :

- si l'on fait l'hypothèse que le problème xyz-DDH est difficile, alors, étant donné le quintuplet $(P, xP, yP, zP, Q) \in \mathbb{G}^5$, personne ne peut décider efficacement si $Q = xyzP$;

- quelqu'un connaissant x, y ou z peut convaincre n'importe qui de l'égalité $Q = xyzP$, en s'engageant dans une preuve d'égalité (interactive ou non, transférable ou non) de deux logarithmes discrets. Supposons par exemple qu'Alice connaisse x. Elle peut alors prouver que
$$\log_{e(yP,zP)}(e(Q,P)) = \log_P(xP);$$

- enfin, après la publication du point xyP (par exemple), tout le monde peut être convaincu de l'égalité $Q = xyzP$: il suffit de s'assurer que
$$e(xyP, zP) = e(Q,P).$$

Cette remarque peut être généralisée en remplaçant $xyzP$ par xyz^tP, avec $t \in \mathbb{N}$. Ceci nous servira pour obtenir une conversion temporelle pour des signatures indéniables.

1.2.2 De nouveaux problèmes

Le problème sur lequel est basé l'anonymat de notre nouveau schéma de signatures indéniables convertibles dans le temps est une variante décisionnelle d'un mélange entre le problème xyz-DDH et k-CAA. La version décisionelle de ce dernier est facile grâce au couplage (comme pour q-SDH). Ce nouveau problème est découpé en 3 étapes, que l'on peut décrire de la façon suivante :

(ℓ, T)-xyz Decisional CAA Problem ((ℓ, T)-xyz-DCAA) : Soit $(\mathbb{G}, +)$ un groupe d'ordre premier q, soit x, y et z des entiers dans $[\![1, q-1]\!]$, et ℓ et T des nombres entiers.

Input : $\left[(x^i P)_{i \in [\![0, 2T-1]\!]}, yP, zP, h, \left(\frac{x^T y}{x + h_k} P, h_k\right)_{k \in [\![1, \ell]\!]}\right]$ in $\mathbb{G}^{2T+2} \times [\![1, q-1]\!] \times (\mathbb{G} \times [\![1, q-1]\!])^{\ell}$, avec $h \notin \{h_1, \ldots, h_\ell\}$

Oracle : à une requête $t^\star \in [\![1, T]\!]$, l'oracle répond $\left[(x^i yP)_{i \in [\![1, t^\star - 1]\!]}, Q\right] \in \mathbb{G}^{t^\star}$

Output : décider si $(x + h)Q = x^{t^\star} yzP$

L'anonymat de notre schéma de signatures indéniables (*i.e.* avec $T = 1$) est lié au problème $(\ell, 1)$-xyz-DCAA. Le lien entre ce dernier et le problème ℓ-CAA de [174] est analogue à celui entre le problème xyz-DDH et le problème CDH.

Définition 39 (**(ℓ, T)-xyz-DCAA**) *Soit k un entier, $\mathcal{G}en$ un générateur de groupes BDH symétrique et soit ℓ et T deux entiers. Soit D un adversaire qui prend en entrée un entier k, $(q, P, \mathbb{G}, \mathbb{H}, e)$ un 5-uple généré par $\mathcal{G}en$ et $\left[(X_i)_{i \in [\![1, 2T-1]\!]}, Y, Z, h, (R_j, h_j)_{j \in [\![1, \ell]\!]}\right]$ dans $\mathbb{G}^{2T+1} \times [\![1, q-1]\!] \times (\mathbb{G} \times [\![1, q-1]\!])^{\ell}$ et qui renvoie un bit. On considère les expériences aléatoires suivantes, où k est un paramètre de sécurité et $r \in \{0, 1\}$:*

$\boxed{\textit{Experiment } \mathbf{Exp}^{(\ell,\mathsf{T})-\mathsf{xyz}-\mathsf{dcaa}-\mathsf{r}}_{\mathcal{G}en,D}(k)}$

$(q, P, \mathbb{G}, \mathbb{H}, e) \leftarrow \mathcal{G}en(k)$
params $\leftarrow (q, P, \mathbb{G}, \mathbb{H}, e)$
$x \xleftarrow{R} [\![1, q-1]\!]$.
Pour i de 1 à $2T - 1$ faire $X_i \leftarrow x^i P$
$(y, z) \xleftarrow{R} [\![1, q-1]\!]^2$, $Y \leftarrow yP$, $Z \leftarrow zP$
$h_1, \ldots, h_\ell, h \xleftarrow{R} [\![1, q-1]\!]$
Pour j de 1 à ℓ faire $R_j = x^T y(x + h_j)^{-1} P$
$t^\star \leftarrow D(k, \text{params}, (X_i)_{i \in [\![1, 2T-1]\!]}, Y, Z, h, (R_j, h_j)_{j \in [\![1, \ell]\!]})$
Pour k de 1 à $t^\star - 1$ faire $Y_k \leftarrow x^k yP$
Si $r = 0$ *alors* $Q \leftarrow x^T yz(x + h)^{-1} P$ *sinon* $Q \xleftarrow{R} \mathbb{G}$
$d \leftarrow D(k, \text{params}, t^\star, (X_i)_{i \in [\![1, 2T-1]\!]}, Y, Z, h, (R_j, h_j)_{j \in [\![1, \ell]\!]}, (Y_k)_{k \in [\![1, t^\star - 1]\!]}, Q)$
Retourner d

*On définit alors l'*avantage *de D pour résoudre le problème $(\ell, T) - xyz$-DCAA comme étant :*

$$\mathbf{Adv}^{(\ell,\mathsf{T})\text{-xyz-dcaa}}_{\mathcal{G}en,D}(k) = \left|\Pr\left[\mathbf{Exp}^{(\ell,\mathsf{T})\text{-xyz-dcaa-0}}_{\mathcal{G}en,D}(k) = 1\right] - \Pr\left[\mathbf{Exp}^{(\ell,\mathsf{T})\text{-xyz-dcaa-1}}_{\mathcal{G}en,D}(k) = 1\right]\right|.$$

Étant donné $\tau \in \mathbb{N}$ et $\varepsilon \in [0, 1]$, le problème (ℓ, T)-xyz-DCAA est dit (k, τ, ε)-difficile si aucun adversaire D, fonctionnant en temps τ, n'a un avantage $\mathbf{Adv}^{(\ell,\mathsf{T})-\mathsf{xyz}-\mathsf{dcaa}}_{\mathcal{G}en,D}(k) \geq \varepsilon$.

Le schéma de signatures dirigées décrit dans la section 3.2.1 repose sur la difficulté du problème ℓ-xyz-décisionnel CAA suivant, qui est en fait le problème $(\ell, 1)$-xyz-DCAA :

Le problème ℓ-xyz-DCAA Soit $(\mathbb{G}, +)$ un groupe d'ordre premier q engendré par P. Soit x_1, x_2, y et z dans $[\![1, q-1]\!]$, et $\ell \in \mathbb{N}$. Étant donné x_1P, x_2P, yP, zP, $(Q, h) \in \mathbb{G} \times [\![1, q-1]\!]$, $\left(\frac{x_1 y}{x_2 + h_k}P, \frac{x_1}{x_2 + h_k}P, h_k\right)_{k \in [\![1,\ell]\!]} \in (\mathbb{G}^2 \times [\![1, q-1)^{\ell}$, avec $h \notin \{h_1, \ldots, h_\ell\}$, le problème ℓ-xyz-DCAA consiste à décider si $Q = \frac{x_1 y z}{x_2 + h}P$.

2

Signatures indéniables

Résumé : Dans ce chapitre, nous nous intéressons aux signatures indéniables introduites par Chaum et van Antwerpen en 1990 dans [57] et largement déclinées par la suite. Nous donnons tout d'abord une définition formelle de ces signatures en l'enrichissant d'une propriété nouvelle, la *conversion temporelle* et proposons dans un premier temps un nouveau schéma de signatures indéniables totalement convertibles, puis une variante possédant la propriété de conversion universelle. Nous développons ces résultats qui ont fait l'objet de la publication à la conférence RSA 2005 d'un article écrit avec Damien Vergnaud [148].

Nous proposons ensuite un schéma de signatures indéniables courtes dont la sécurité est prouvée dans le modèle standard : ce résultat a été obtenu en collaboration avec Damien Vergnaud et soumis [147].

Sommaire

2.1 Introduction **85**
2.1.1 Définition 87
2.1.2 Modèle de sécurité 88
2.2 CUSBM : nouvelles signatures indéniables convertibles **92**
2.3 TSCUSBM : signatures indéniables convertibles dans le temps. . . **92**
2.4 Preuves de sécurité **93**
2.4.1 Inforgeabilité 94
2.4.2 Anonymat 98
2.5 Signatures indéniables courtes dans le modèle standard **100**
2.5.1 Preuve d'égalité de logarithme discret 101
2.5.2 USWRO : description 103
2.5.3 Preuves de sécurité 103
2.6 Conclusion **108**

2.1 Introduction

Le concept de *signature indéniable* a été introduit en 1989 par Chaum et van Antwerpen [57], motivé par la volonté de pouvoir contrôler la vérification de signatures numériques. En effet, les signatures "classiques" ont la propriété (fondamentale) d'être universellement vérifiables, c'est-à-dire que toute personne en leur possession peut se convaincre seule de leur validité ou non. Le champ d'applications numériques des signatures étant particulièrement vaste, dans de nombreuses situations, ce sont des propriétés plus subtiles que l'on demande aux signatures. Dans certains cas, il devient important que la vérification d'une signature soit contrôlée (par exemple par le signataire), et que la conviction apportée à un vérificateur de leur validité ou non ne puisse être transmise à une tierce partie. L'exemple proposé par Chaum dans [54] est celui du contrôle des licences de logiciel. En effet, un fabricant de logiciels, signant un de ses produits, souhaite que seules les personnes ayant acheté le logiciel puissent être convaincues, grâce à cette signature, que le logiciel est authentique et non modifié. De plus si une personne en possession d'un logiciel signé le distribue sans autorisation, l'acquéreur ne pourra se convaincre de l'authenticité du logiciel à partir de la seule signature du fabriquant, même en interagissant avec cette personne.

Ainsi, une signature indéniable est une signature qui nécessite une interaction avec le signataire pour être vérifiée : à travers un protocole de confirmation avec un vérifieur, le signataire prouve la validité d'une signature, et à travers un protocole de démenti, il prouve l'invalidité d'une autre. Il ne pourra pas au cours du protocole de démenti prouver qu'une signature n'est pas valide s'il l'a réellement produite (d'où le qualificatif d'indéniable).

Le premier schéma proposé par Chaum et van Antwerpen [57] souffrait du fait que les protocoles de confirmation et de démenti de la validité d'une signature indéniable n'étaient pas zero-knowledge. Chaum, un an plus tard, propose une version zero-knowledge de ces protocoles dans [54]. En 1991, Boyar, Chaum, Damgård et Pedersen dans [42] étendent le concept en proposant une possibilité de conversion (individuelle - *individual conversion* - ou totale - *universal conversion* -) des signatures. La conversion individuelle consiste à modifier une signature indéniable pour qu'elle soit vérifiable par tout le monde : c'est une transformation qui rend cette signature ordinaire. La conversion totale rend *toutes* les signatures vérifiables au sens des signatures classiques. Il n'est alors plus nécessaire d'interagir avec le signataire. Ces conversions sont importantes quand il est utile qu'une signature devienne *a posteriori* universellement vérifiable, par exemple après une certaine période de temps. Le schéma de signatures indéniables convertibles de Boyar *et al.*, basé sur El Gamal, a été cryptanalysé en 1996 par Michels, Petersen et Horster dans [168], qui ont proposé une version réparée, avec une sécurité heuristique. Ces signatures ont par la suite inspiré de nombreux chercheurs, et des schémas de signatures ont fleuri : basés sur El Gamal par Damgård et Pedersen dans [70], sur Schnorr par Michels et Stadler [169], sur RSA par Galbraith et Mao [95], Galbraith, Mao et Paterson [96], par Gennaro, Krawczyk et Rabin dans [101], sur des idéaux dans des ordres quadratiques par Biehl, Paulus and Takagi [27], sur des calculs de caractères par Monnerat et Vaudenay [176]. Libert et Quisquater ont proposé un schéma de signatures indéniables basé sur des applications bilinéaires dans [155] (qui ne possède pas la conversion totale).

À Eurocrypt'91 [192], Pedersen propose un schéma à seuil : le vérificateur a partagé son secret entre plusieurs personnes. Pour vérifier une signature, lorsque le signataire n'est pas en mesure d'interagir avec le vérificateur, celui-ci peut faire appel à un certain nombre des personnes en possession d'une part du secret. À la même conférence, Fujioka, Okamoto et Ohta [92] proposent un seul et même protocole pour la confirmation et le démenti d'une signature (en développant la notion de bi-preuve interactive). On peut mentionner aussi les *signatures non-transitives* présentées dans le papier d'Okamoto et Ohta [187], qui sont proches des signatures indéniables. Grâce à ces signatures, seul le signataire peut prouver à un vérificateur (au cours d'une interaction) qu'un message a bien été émis par lui, et sans produire de signature proprement dite. Le vérificateur ne peut pas, lui, transmettre sa conviction et prouver l'origine du message à un tiers. Pour régler les éventuels problèmes de litige, Park, Kim, An et Won ont introduit dans [191] des signatures undeniables qui contiennent un engagement des aléas utilisés pour signer, vers une autorité.

Dans tous ces protocoles, la conversion totale consiste essentiellement à révéler une partie de la clé secrète. Cette façon de convertir implique que la transformation des signatures indéniables en signatures classiques est définitive, c'est-à-dire que les signatures passées *et* futures deviennent universellement vérifiables. De plus, cette façon de faire compromet la clé publique, et nécessite donc, en particulier dans les infrastructures à clés publiques, la mise en œuvre de la certification d'une nouvelle clé pour pouvoir à nouveau générer des signatures indéniables, et donc l'enregistrement de nombreuses clés. Aussi, nous proposons avec Damien Vergnaud dans [148] une définition du concept de *conversion dans le temps* des signatures, qui offre la possibilité de convertir individuellement des signatures sur une certaine période de temps, ou de convertir toutes les signatures jusqu'à une certaine période de temps. Une définition formelle de cette nouvelle conversion est proposée dans la section 2.1.1. Ensuite, nous proposons un nouveau schéma de signatures indéniables qui s'étend naturellement à un schéma dans lequel les signatures sont convertibles dans le temps. Dans la dernière partie, nous proposons un schéma de signatures indéniables dont la taille est petite, et dont la sécurité est prouvée sans faire d'hypothèse idéale sur les fonctions de hachage. Ce travail avec Damien Vergnaud a fait l'objet d'une soumission [147].

2.1.1 Définition

Une nouvelle propriété : la conversion temporelle Dans cette section, nous définissons formellement le concept de signatures indéniables convertibles dans le temps (*Time Selective Convertible Undeniable Signatures* - TSCUS). Nous conservons le terme historique de signatures indéniables, bien qu'il ne soit pas particulièrement pertinent (on pourrait les qualifier de "non-vérifiables", ou "non-transférables").

Définition 40 (Schéma de signatures indéniables convertibles dans le temps) *Soit k un entier. Un* schéma de signatures indéniables convertibles dans le temps TSCUS *de paramètre de sécurité k est composé des différents algorithmes et protocoles suivants :*

- génération des paramètres publics TSCUS.Setup *: algorithme probabiliste prenant en entrée un entier k et renvoyant* les paramètres publics *;*
- génération de clés TSCUS.KeyGen *: algorithme probabiliste prenant en entrée un entier k, des paramètres publics et renvoyant* une paire de clés publique et secrète (pk, sk) *et un* nombre de périodes de temps $T \in \mathbb{N}$ *;*
- signature TSCUS.Sign *: algorithme déterministe ou probabiliste prenant en entrée un entier k, des paramètres publics, une clé secrète sk, un message $m \in \{0,1\}^*$, un nombre de périodes de temps T, un entier $t \in [\![1, T]\!]$ et renvoyant σ, une* signature indéniable convertible à la période de temps t de m *;*
- protocoles de confirmation/démenti TSCUS.{Confirm, Deny} *: protocoles prenant en entrée un entier k, des paramètres publics, une paire de clés (pk, sk) (et eventuellement d'autres paires de clés), un message $m \in \{0,1\}^*$, un nombre de périodes de temps T, un entier $t \in [\![1, T]\!]$, une suite de bits σ. Leur sortie est une* preuve (éventuellement non-interactive) non transférable *que σ est, ou non, une signature indéniable convertible pour la période t de m par rapport à la clé publique pk ;*

- génération de reçu de conversion individuelle TSCUS.IReceipt *: algorithme déterministe prenant en entrée un entier k, des paramètres publics, une clé secrète sk, un message $m \in \{0,1\}^*$, une suite de bits σ, un nombre de périodes de temps T et un entier $t \in [\![1, T]\!]$, et renvoyant un* reçu de conversion individuelle $\tilde{\sigma}$ *;*
- vérification d'une signature et de son reçu de conversion individuelle TSCUS.IVerify *: algorithme*

déterministe prenant en entrée un entier k, des paramètres publics, une clé publique pk, un message $m \in \{0,1\}^*$, un nombre de périodes de temps T, un entier $t \in [\![1,T]\!]$, deux suites de bits σ et $\tilde{\sigma}$, et vérifiant que $\tilde{\sigma}$ est un reçu de conversion individuelle valide de σ pour la clé publique pk et la période de temps t. Si c'est le cas, cet algorithme teste si σ est une signature indéniable valide pour la période t de m par rapport à la clé publique pk, sinon, il renvoie `Erreur`*;*

- génération de reçu conversion totale *TSCUS.TReceipt : algorithme déterministe prenant en entrée un entier k, des paramètres publics, une clé secrète sk, un nombre de périodes de temps T, un entier $t \in [\![1,T]\!]$ et renvoyant un* reçu de conversion totale *$\mathcal{I}_t$;*
- vérification d'une signature et d'un reçu de conversion totale *TSCUS.TVerify : algorithme déterministe prenant en entrée un entier k, des paramètres publics, une clé publique pk, un message $m \in \{0,1\}^*$, un nombre de périodes de temps T, un couple d'entiers $(t,t') \in [\![1,T]\!]^2$ avec $t' \leq t$, deux suites de bits σ et $\mathcal{I}_t$, et vérifiant que $\mathcal{I}_t$ est un reçu de conversion totale valide pour la période de temps t par rapport à la clé pk. Si c'est le cas, il teste si σ est une signature indéniable valide pour la période de temps t' de m par rapport à la clé pk, sinon, il renvoie* `Erreur`*;*

Cette définition généralise la définition classique des signatures indéniables convertibles, puisqu'on la retrouve ainsi :

Définition 41 (Schéma de signatures indéniables totalement convertibles) *Étant donné un entier k, un* schéma de signatures indéniables totalement convertibles *de paramètre de sécurité k est un schéma de signatures indéniables convertibles dans le temps de paramètre de sécurité k dont l'algorithme de génération de clés fixe systématiquement le nombre de périodes de temps T à 1.*

2.1.2 Modèle de sécurité

Dans cette partie, nous donnons les définitions formelles des notions de sécurité des signatures indéniables, qui tiennent compte de la possibilité de convertir des signatures dans le temps. On note

$$\mathsf{TSCUS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Confirm}, \mathsf{Deny}, \mathsf{IReceipt}, \mathsf{IReceipt}, \mathsf{TReceipt}, \mathsf{TReceipt})\,.$$

2.1.2.1 Consistance des vérifications

Définition 42 (Consistance de la vérification d'un reçu de conversion individuelle) *La vérification des reçus de conversion individuelle est dite* consistante *si, pour tout $k \in \mathbb{N}$, pour tout $\mathsf{params} \in \{\mathit{TSCUS.Setup}(k)\}$, pour toute paire de clés $(pk_A, sk_A, T) \in \{\mathit{TSCUS.SKeyGen}(k, \mathsf{params})\}$, pour tout message $m \in \{0,1\}^*$, pour tout $t \in [\![1,T]\!]$,*

$$\mathit{TSCUS.IVerify}\big(k, \mathsf{params}, pk_A, m, T, t, \sigma, \tilde{\sigma},\big) = 1$$

si et seulement si

$$\sigma \in \big\{\mathit{TSCUS.Sign}(k, \mathsf{params}, sk_A, m, T, t)\big\} \text{ et } \tilde{\sigma} = \mathit{TSCUS.IReceipt}(k, \mathsf{params}, sk_A, m, \sigma, T, t).$$

Définition 43 (Consistance de la vérification d'un reçu de conversion totale) *La vérification des reçus de conversion totale est dite* consistante *si, pour tout $k \in \mathbb{N}$, pour tout $\mathsf{params} \in$*

$\{\textsf{TSCUS.Setup}(k)\}$, *pour toute paire de clés* $(pk_A, sk_A, T) \in \{\textsf{TSCUS.SKeyGen}(k, \textsf{params})\}$, *pour tout message* $m \in \{0,1\}^*$, *pour tout* $(t, t') \in [\![1, N]\!]^2$ *vérifiant* $t' \leq t$,

$$\textsf{TSCUS.TVerify}\big(k, \textsf{params}, pk_A, m, T, (t, t'), \sigma, \mathcal{I}_t\big) = 1$$

si et seulement si

$$\sigma \in \{\textsf{TSCUS.Sign}(k, \textsf{params}, sk_A, m, T, t)\} \text{ et } \mathcal{I}_t = \textsf{TSCUS.TReceipt}(k, \textsf{params}, sk_A, T, t).$$

2.1.2.2 Inforgeabilité

La sécurité des signatures indéniables repose sur la notion standard définie dans [116] par Goldwasser, Micali et Rivest. Pour des signatures indéniables la différence notable réside dans le fait que l'attaquant, durant son attaque à messages choisis, a accès à un oracle de génération de reçus de conversion Υ, un oracle de confirmation/démenti Ξ, en plus de l'accès à l'oracle de signature Σ (et, éventuellement, à un oracle aléatoire $\mathcal{H}$). Il peut donc participer à des protocoles de confirmation/démenti si ceux-ci sont interactifs, et obtenir les informations de conversion sur les signatures de son choix. Pour les signatures convertibles dans le temps, l'attaquant a aussi accès à la totalité des reçus de conversion totale $\mathcal{T}_t$, pour $t \in [\![1, T]\!]$. La forge $(m^\star, \sigma^\star, t^\star)$ produite par l'attaquant ne doit bien sûr pas provenir d'une requête $m^\star$ à l'oracle de signature pour la période de temps $t^\star$. Cependant, il peut interroger l'oracle de signature pour obtenir des signatures de $m^\star$ pour n'importe quel autre période de temps.

Définition 44 (Inforgeabilité) *Soit k un entier et* TSCUS *un schéma de signatures indéniables convertibles dans le temps, et $\mathcal{A}$ un adversaire* EF-CMA *contre* TSCUS. *Considérons les expériences aléatoires suivantes, où k est un paramètre de sécurité :*

$\boxed{\textit{Experiment } \mathbf{Exp}^{\mathsf{ef-cma}}_{\mathsf{TSCUS},\mathcal{A}}(k)}$
params $\leftarrow \textsf{TSCUS.Setup}(k)$
$(pk, sk, T) \leftarrow \textsf{TSCUS.KeyGen}(k, \text{params})$
Pour t de 1 à T faire $\mathcal{I}_t \leftarrow \textsf{TSCUS.TReceipt}(k, \textsf{params}, sk, T, t)$
$(m^\star, \sigma^\star, t^\star) \leftarrow \mathcal{A}^{\mathcal{H},\Sigma,\Upsilon,\Xi}(k, \text{params}, pk, \{\mathcal{I}_t\}_{t \in [\![1,T]\!]})$
Retourner $\textsf{TSCUS.TVerify}(k, \textsf{params}, pk, m^\star, T, (T, t^\star), \sigma^\star, \mathcal{I}_T)$

On définit alors le succès *de l'adversaire $\mathcal{A}$ comme étant*

$$\mathbf{Succ}^{\textsf{ef-cma}}_{\textsf{TSCUS},\mathcal{A}}(k) = \Pr\left[\mathbf{Exp}^{\textsf{ef-cma}}_{\textsf{TSCUS},\mathcal{A}}(k) = 1\right].$$

Étant donné $\tau \in \mathbb{N}$ et $\varepsilon \in [0, 1]$, le schéma TSCUS *est dit (k, τ, ε)-*EF-CMA *sûr si aucun adversaire* EF-CMA *$\mathcal{A}$ fonctionnant en temps τ, n'a un succès $\mathbf{Succ}^{\textsf{ef-cma}}_{\textsf{TSCUS},\mathcal{A}}(k) \geq \varepsilon$.*

2.1.2.3 Anonymat/Invisibilité

Les notions d'anonymat et d'invisibilité pour les signatures indéniables ont été formalisées par Galbraith et Mao dans [95]. La notion d'invisibilité avait été proposée par Camenisch et Michels en 2000 [49] pour des signatures à confirmeur désigné. Informellement, un schéma de signatures indéniables est invisible si un attaquant n'est pas capable de déterminer si couple message/signature indéniable est valide pour un utilisateur donné. Dans [95], Galbraith et Mao proposent une variante invisible du schéma de Gennaro, Krawcyk et Rabin [101], basé sur RSA, qui ne l'était pas. Ils ont proposé aussi la notion d'anonymat pour les signatures de ce type (dans un contexte multi-utilisateur), et ont montré que cette notion était essentiellement équivalente à celle d'invisibilité.

Un schéma de signatures indéniables est anonyme si un attaquant, en possession d'un couple message/signature indéniable valide, et face à deux signataires potentiels, n'est pas capable de dire lequel des deux a produit la signature. Il est invisible, si l'attaquant qui possède la clé publique d'un signataire, n'est pas capable de savoir si une suite de bits donnée est une signature de cette personne, ou si c'est un élément quelconque de l'ensemble de toutes les signatures possibles.

L'adversaire $\mathcal{A}$ considéré pour ces deux notions s'engage dans une attaque à messages choisis. Il fonctionne en deux étapes - $\mathcal{A} = (\texttt{find}, \texttt{guess})$ -, et a accès, pendant ces deux étapes aux oracles suivants :

– un oracle de signatures Σ,
– un oracle de génération de reçus individuels Υ,
– un oracle de confirmation/démenti Ξ,
– éventuellement un oracle aléatoire $\mathcal{H}$.

Dans sa deuxième phase, l'attaquant reçoit également les reçus universels jusqu'à sa période cible $t^\star$, choisie à l'issue de sa première phase : $\mathcal{I}_1, \ldots, \mathcal{I}_{t^\star}$. La seule restriction imposée à l'attaquant est la suivante : il ne doit pas demander une confirmation, un démenti ou un reçu individuel à l'oracle correspondant à propos de la signature cible. En particulier lorqu'il a accès aux oracles pour deux signataires potentiels, il n'interroge aucun des oracles précédents leur correspondant sur la signature cible.

Les définitions formelles sont alors les suivantes :

Définition 45 (Anonymat) *Soit k un entier et* TSCUS *un schéma de signatures indéniables convertibles dans le temps, et $\mathcal{A}$ un adversaire* Ano-CMA *contre* TSCUS*. On considère les expériences aléatoires suivantes, pour $r \in \{0,1\}$, où k est un paramètre de sécurité :*

$\boxed{\textit{Experiment } \mathbf{Exp}^{\textsf{ano-cma}-r}_{\textsf{TSCUS},\mathcal{A}}(k)}$
$\text{params} \leftarrow \textsf{TSCUS.Setup}(k)$
$(pk_0, sk_0, T_0) \leftarrow \textsf{TSCUS.KeyGen}(k, \text{params})$
$(pk_1, sk_1, T_1) \leftarrow \textsf{TSCUS.KeyGen}(k, \text{params})$
$(m^\star, t^\star, s) \leftarrow \mathcal{A}^{\mathcal{H},\Sigma_0,\Sigma_1,\Upsilon_0,\Upsilon_1,\Xi_0,\Xi_1}(k, \texttt{find}, \text{params}, pk_0, T_0, pk_1, T_1)$ *avec* $t^\star \leq \min(T_0, T_1)$
$\sigma^\star \leftarrow \textsf{TSCUS.Sign}(k, \text{params}, sk_r, m, T_r, t^\star)$
Pour t de 1 à $t^\star - 1$ faire
  $\mathcal{I}_t^0 \leftarrow \textsf{TSCUS.TReceipt}(k, \text{params}, sk_0, T_0, t)$
  $\mathcal{I}_t^1 \leftarrow \textsf{TSCUS.TReceipt}(k, \text{params}, sk_1, T_1, t)$
$d \leftarrow \mathcal{A}^{\mathcal{H},\Sigma_0,\Sigma_1,\Upsilon_0,\Upsilon_1,\Xi_0,\Xi_1}(k, \texttt{guess}, s, \text{params}, pk_0, T_0, pk_1, T_1, m^\star, \sigma^\star, t^\star, \{\mathcal{I}_t^0, \mathcal{I}_t^1\}_{j \in [\![1, t^\star-1]\!]})$
Retourner d

*On définit alors l'*avantage *de l'adversaire $\mathcal{A}$ comme étant*

$$\mathbf{Adv}^{\textsf{ano-cma}}_{\textsf{TSCUS},\mathcal{A}}(k) = \left|\Pr\left[\mathbf{Exp}^{\textsf{ano-cma}-1}_{\textsf{TSCUS},\mathcal{A}}(k) = 1\right] - \Pr\left[\mathbf{Exp}^{\textsf{ano-cma}-0}_{\textsf{TSCUS},\mathcal{A}}(k) = 1\right]\right|.$$

Étant donné $\tau \in \mathbb{N}$ et $\varepsilon \in [0,1]$, le schéma TSCUS *est dit (k, τ, ε)-*Ano-CMA *sûr, si aucun adversaire* Ano-CMA *$\mathcal{A}$ fonctionnant en temps τ n'a un avantage $\mathbf{Adv}^{\textsf{ano-cma}}_{\textsf{TSCUS},\mathcal{A}}(k) \geq \varepsilon$.*

Définition 46 (Invisibilité) *Soit k un entier et TSCUS un schéma de signatures indéniables convertibles dans le temps de paramètre de sécurité k. On note $\mathcal{S}ign$ l'ensemble des signatures possibles pour TSCUS. Soit $\mathcal{A}$ un adversaire $\mathsf{Inv\text{-}CMA}$ contre TSCUS. Considérons l'expérience aléatoire suivante, pour un $r \in \{0,1\}$:*

$$
\begin{array}{l}
\boxed{\textit{Experiment } \mathbf{Exp}^{\textsf{inv-cma-}r}_{\mathsf{TSCUS},\mathcal{A}}(k)} \\
\text{params} \leftarrow \mathsf{TSCUS.Setup}(k) \\
(pk, sk, T) \leftarrow \mathsf{TSCUS.KeyGen}(k, \text{params}) \\
(m^\star, t^\star, s) \leftarrow \mathcal{A}^{\Sigma,\Xi,\Upsilon}(k, \mathsf{find}, pk, T) \ \textit{avec } t^\star \leq T \\
\textit{Si } r = 0 \textit{ alors } \sigma^\star \leftarrow \mathsf{TSCUS.Sign}(k, \text{params}, sk, m^\star, T, t^\star) \\
\textit{Si } r = 1 \textit{ alors } \sigma^\star \xleftarrow{R} \mathcal{S}ign \\
\textit{Pour } t \textit{ de } 1 \textit{ à } t^\star - 1 \textit{ faire} \\
\quad \mathcal{I}_t \leftarrow \mathsf{TSCUS.TReceipt}(k, \text{params}, sk, T, t) \\
d \leftarrow \mathcal{A}^{\Sigma,\Xi,\Upsilon}(k, \mathsf{guess}, s, pk, T, m^\star, \sigma^\star, \{\mathcal{I}_t\}_{t \in [\![1, t^\star - 1]\!]}) \\
\textit{Retourner } d
\end{array}
$$

*On définit alors l'*avantage *de l'attaquant $\mathcal{A}$ comme étant*

$$\mathbf{Adv}^{\mathsf{inv-cma}}_{\mathsf{TSCUS},\mathcal{A}}(k) = \left| \Pr\left[\mathbf{Exp}^{\mathsf{inv-cma-1}}_{\mathsf{TSCUS},\mathcal{A}}(k) = 1\right] - \Pr\left[\mathbf{Exp}^{\mathsf{inv-cma-0}}_{\mathsf{TSCUS},\mathcal{A}}(k) = 1\right] \right|.$$

Étant donné $\tau \in \mathbb{N}$ et $\varepsilon \in [0,1]$, le schéma TSCUS est dit (k, τ, ε)-$\mathsf{Inv\text{-}CMA}$ sûr si aucun adversaire $\mathcal{A}$ fonctionnant en temps τ, n'a un avantage $\mathbf{Adv}^{\mathsf{inv-cma}}_{\mathsf{TSCUS},\mathcal{A}}(k) \geq \varepsilon$.

2.1.2.4 Non-transférabilité

La notion de non-transférabilité (de la conviction sur la validité/invalidité d'une signature) signifie qu'un utilisateur qui participe à une vérification d'une signature indéniable, même s'il triche, n'obtient pas assez d'informations pour convaincre une tierce partie de la validité/invalidité de la signature. Il ne peut pas transférer sa conviction à quelqu'un. Suivant la définition de Camenisch et Michels [49] de la non-transférabilité des signatures à confirmeur, la définition devient dans le cas des signatures indéniables convertibles dans le temps :

Définition 47 (Non transférabilité) *Soit k un entier et TSCUS un schéma de signatures indéniables convertibles dans le temps de paramètre de sécurité k. Soit $\mathcal{A}$ un adversaire $\mathsf{NT\text{-}CMA}$ contre TSCUS.*

Soit τ un entier et ε un réel dans $[0,1]$. Le schéma TSCUS est dit (k, τ, ε)-$\mathsf{NT\text{-}CMA}$ s'il existe un simulateur $\mathsf{Simulator}$ fonctionnant en temps polynomial, pour lequel, si l'on considère l'expérience aléatoire suivante, pour un $r \in \{0,1\}$:

$$
\begin{array}{l}
\boxed{\textit{Experiment } \mathbf{Exp}^{\textsf{nt-cma-}r}_{\mathsf{TSCUS},\mathcal{A}}(k)} \\
\text{params} \leftarrow \mathsf{TSCUS.Setup}(k) \\
(pk, sk, T) \leftarrow \mathsf{TSCUS.KeyGen}(k, \text{params}) \\
(m^\star, \sigma^\star, t^\star) \leftarrow \mathcal{A}^{\mathcal{H},\Sigma,\Upsilon,\Xi}(\text{params}, pk, T) \\
\textit{Si } r = 0 \textit{ faire } \mathsf{Proof} = \Xi(m^\star, \sigma^\star, t^\star) \\
\textit{Si } r = 1 \textit{ faire } \mathsf{Proof} = \mathsf{Simulator}(m^\star, \sigma^\star, t^\star) \\
d \leftarrow \mathcal{A}^{\mathcal{H},\Sigma,\Upsilon,\Xi}(\mathsf{guess}, \text{params}, pk, m^\star, \sigma^\star, t^\star, \mathsf{Proof}) \\
\textit{Retourner } d
\end{array}
$$

et si l'on définit l'avantage de l'adversaire $\mathcal{A}$, comme étant

$$\mathbf{Adv}^{\mathsf{nt-cma}}_{\mathsf{TSCUS},\mathcal{A}}(k) = \left|\Pr\left[\mathbf{Exp}^{\mathsf{nt-cma-1}}_{\mathsf{TSCUS},\mathcal{A}}(k) = 1\right] - \Pr\left[\mathbf{Exp}^{\mathsf{nt-cma-0}}_{\mathsf{TSCUS},\mathcal{A}}(k) = 1\right]\right|,$$

aucun adversaire NT-CMA $\mathcal{A}$ fonctionnant en temps τ, n'a un avantage $\mathbf{Adv}^{\mathsf{nt\text{-}cma}}_{\mathsf{TSCUS},\mathcal{A}}(k) \geq \varepsilon$.

2.2 CUSBM : nouvelles signatures indéniables convertibles

Dans cette section, nous proposons un protocole de signatures indéniables convertibles CUSBM (pour Convertible Undeniable Signatures based on Bilinear Maps) qui s'inspire à la fois des signatures courtes de Boneh, Lynn, et Shacham [38] et des signatures courtes de Zhang, Safavi-Naini et Susilo [247, 248], similaires à celle de Boneh et Boyen [34].

La figure 2.1 présente la génération des paramètres publics et des clés ainsi que la signature, la figure 2.2 fournit les protocoles de confirmation et de démenti, et la figure 2.3 décrit les protocoles de génération des reçus de conversion individuelle et totale, ainsi que les vérifications correspondantes.

Comme dans le schéma de Michels et Stadler de SAC'97 [169], le signataire peut non seulement convertir des signatures valides, mais il peut aussi produire un reçu attestant de la non-validité d'une signature. Malheureusement, cela ne peut se faire par la même opération de publication du point $a_2H(m||r)$. En effet, si ce point est systématiquement publié, sans que le signataire ne vérifie la signature préalablement, un attaquant en possession d'une signature potentielle (σ, r) sur m, peut demander une information de confirmation sur $m, (\sigma', r)$, où σ' est un élément quelconque de $\mathbb{G}$, et obtient ainsi $a_2H(m||r)$, qui lui permet de vérifier la signature σ.

Le coût d'une signature est essentiellement celui d'une inversion dans le corps de base, auquel s'ajoute celui d'une multiplication scalaire sur la courbe. La taille des signatures est celle d'un point sur une courbe et de quelques bits d'aléa. Les protocoles de confirmation et démenti sont des preuves classiques d'égalité et d'inégalité de logarithmes discrets. Elles doivent être non-transférables : par exemple des preuves à vérificateurs désignés [125]. Les générations de reçus ne coûtent qu'une multiplication scalaire, ou la transformation d'une preuve interactive en signature. La vérification de ces reçus, lorsque ce sont des points, se fait au prix de 2 calculs de couplages.

Les réductions aux problèmes sous-jacents exhibées dans les preuves de sécurité étant fines, ce nouveau schéma est le plus performant des schémas de signatures convertibles existants.

2.3 TSCUSBM : signatures indéniables convertibles dans le temps.

Le schéma précédent s'étend de façon tout à fait naturelle à un schéma de signatures indéniables convertibles dans le temps TSCUSBM (pour Time Selective Convertible Undeniable Signatures based on Bilinear Maps). Pour obtenir une signature pour la période t, il suffit, dans l'algorithme de signature de CUSBM, d'élever a_2 à la puissance t au numérateur. Les changements qui en découlent dans les autres algorithmes sont immédiats. Les coûts des générations des reçus et de leur vérification croissent avec t, mais une utilisation séquentielle du schéma permet de le rendre beaucoup plus performant.

Les figures 2.4, 2.5 et 2.6 présentent respectivement la génération des paramètres publics et des clés ainsi que la signature, les protocoles de confirmation et de démenti, et les protocoles de génération des reçus individuels et universels ainsi que leur vérification. Les protocoles de confirmation et démenti sont des preuves assez peu efficaces que l'on peut trouver dans l'article de Camenisch et Stadler [50].

Génération des paramètres publics : CUSBM.Setup

• k est un paramètre de sécurité, et $n_r \in \mathbb{N}$ est fixé
• $(q, P, \mathbb{G}, \mathbb{H}, e)$ est généré par BDHGen(k)
• Soit $[\{0,1\}^* \times \{0,1\}^{n_r} \longrightarrow \mathbb{G}]$ une famille de fonctions de hachage et H un élément pris au hasard dans cette famille.
• Soit $[\{0,1\}^* \times \{0,1\}^{n_r} \longrightarrow [\![1, q-1]\!]]$ une famille de fonctions de hachage et h un élément aléatoire de cette famille.

Génération des clés : CUSBM.KeyGen

• Alice tire au hasard $(a_1, a_2) \in [\![1, q-1]\!]^2$ qui est la clé secrète ;
• sa clé publique est (P_1, P_2) avec $P_1 = a_1 P$ et $P_2 = a_2 P$.

Signature : CUSBM.Sign

• Étant donné un message m, ses clés secrètes de signature a_1 et a_2, Alice tire au hasard un élément $r \in \{0,1\}^{n_r}$ et calcule $t = a_2 + h(m||r)$. Si $t = 0$, elle choisit un nouveau r, et sinon elle pose

$$\sigma = \frac{a_1 a_2}{a_2 + h(m||r)} H(m||r).$$

• La signature du message m est (σ, r).

Fig. 2.1 – CUSBM : description de la signature

Protocoles de confirmation et de démenti :

– CUSBM.Confirm
  • Étant donné un message m et une signature (σ, r), Alice confirme celle-ci grâce à la preuve suivante :

$$ZKP(a_2 : e(\sigma, P_2 + h(m||r)P) = e(H(m||r), P_1)^{a_2} \wedge P_2 = a_2 P).$$

– CUSBM.Deny
  • Étant donné un message m et une signature invalide (σ, r), Alice dément celle-ci grâce à la preuve suivante :

$$ZKP(a_2 : e(\sigma, P_2 + h(m||r)P) \neq e(H(m||r), P_1)^{a_2} \wedge P_2 = a_2 P).$$

Fig. 2.2 – CUSBM : description des protocoles de confirmation et démenti

2.4 Preuves de sécurité

Dans cette partie, nous prouvons la sécurité des deux schémas de signatures indéniables, les preuves de sécurité du schéma CSUBM étant des conséquences immédiates de celles du schéma

Génération des reçus de conversion et vérification :

– CUSBM.IReceipt
• Étant donné un message m et une signature potentielle (σ, r), Alice vérifie que la signature est valide. Si c'est le cas, l'information de vérification pour (σ, r) est le point
$$\tilde{\sigma} = a_2 H(m||r),$$
sinon, elle donne
$$\tilde{\sigma} = NIZKP(a_2 : e(\sigma, P_2 + h(m||r)P) \neq e(H(m||r), P_1)^{a_2} \wedge P_2 = a_2 P).$$

– CUSBM.IVerify
• Étant donné un message m et une signature potentielle (σ, r) accompagnée de l'information de vérification $\tilde{\sigma} \in \mathbb{G}$, si
$$e(\tilde{\sigma}, P) = e(P_2, H(m||r))$$
(σ, r) est valide si et seulement si
$$e(\sigma, P_2 + h(m||r)P) = e(\tilde{\sigma}, P_1).$$
Si $\tilde{\sigma} \notin \mathbb{G}$, il faut vérifier la preuve non-interactive.

– CUSBM.TReceipt
• Alice publie le point $I = a_1 a_2 P$ comme reçu de conversion totale.

– CUSBM.TVerify
• Le reçu de conversion totale est d'abord vérifié en s'assurant que
$$e(P_1, P_2) = e(I, P)$$
• la validité de toute signature (σ, r) d'un message m se vérifie ensuite grâce à
$$e(\sigma, P_2 + h(m||r)P) = e(H(m||r), I).$$

FIG. 2.3 – CUSBM : description des algorithmes de conversion et vérification

de signatures convertibles dans le temps TSCUSBM.

2.4.1 Inforgeabilité

Le théorème suivant et son corollaire relient l'inforgeabilité des schémas de signatures indéniables présentés dans la section précédente au problème Diffie-Hellman calculatoire. Les preuves sont conduites dans le modèle de l'oracle aléatoire. Cependant, on pourrait ne pas faire d'hypothèse idéale sur la fonction de hachage h, en utilisant la technique de Boneh-Boyen dans [34].

Théorème 13 (Inforgeabilité de TSCUSBM) *Soit BDHGen un générateur de groupes BDH symétrique et soit TSCUSBM le schéma de signatures indéniables convertibles dans le temps associé. Soit $f_r : \mathbb{N} \longrightarrow \mathbb{N}$ une fonction. Dans le modèle de l'oracle aléatoire, pour tout adversaire*

Génération des paramètres publics : $\mathsf{TSCUSBM.Setup} = \mathsf{CUSBM.Setup}$

Génération des clés : $\mathsf{TSCUSBM.KeyGen}$

• Alice tire au hasard $(a_1, a_2) \in [\![1, q-1]\!]^2$ qui est la clé secrète,
• elle choisit le nombre de périodes de temps $T \in \mathbb{N}$.
• Sa clé publique est (P_1, P_2, T), avec $P_1 = a_1 P$ et $P_2 = a_2 P$.

Signature : $\mathsf{TSCUSBM.Sign}$

• Étant donné un message m, une période de temps $t \leq T$ et les clés secrètes de signature a_1 et a_2, Alice tire au hasard un elément $r \in \{0,1\}^{n_r}$ et calcule le point

$$\sigma = \frac{a_1 a_2^t}{a_2 + h(m||r)} H(m||r).$$

• La signature du message m, associée à la période t, est (σ, r, t).

FIG. 2.4 – TSCUSBM : description de la signature

Protocoles de confirmation et de démenti :

– $\mathsf{TSCUSBM.Confirm}$
  • Étant donné un message m et une signature (σ, r, t), Alice confirme celle-ci grâce à la preuve interactive de connaissance suivante :

$$ZKP(a_2 : e(\sigma, P_2 + h(m||r)P) = e(H(m||r), P_1)^{a_2^t} \wedge P_2 = a_2 P).$$

– $\mathsf{TSCUSBM.Deny}$
  • Étant donné un message m et une signature invalide (σ, r, t), Alice dément celle-ci grâce à la preuve interactive de connaissance suivante :

$$ZKP(a_2 : e(\sigma, P_2 + h(m||r)P) \neq e(H(m||r), P_1)^{a_2^t} \wedge P_2 = a_2 P).$$

FIG. 2.5 – TSCUSBM : description des protocoles de confirmation et démenti

EF-CMA $\mathcal{A}$ contre TSCUSBM, de paramètre de sécurité k, ayant un succès $\mathbf{Succ}^{\mathsf{ef-cma}}_{\mathsf{TSCUSBM},\mathcal{A}}(k) = \varepsilon$ en temps τ, et pouvant faire $q_{\mathcal{H}}$ requêtes à l'oracle aléatoire, q_{Σ} requêtes à l'oracle de signature et q_{Υ} requêtes à l'oracle de génération de reçu de conversion, il existe un adversaire $\mathcal{D}$ contre CDH, de succès $\mathbf{Succ}^{\mathsf{cdh}}_{\mathsf{BDHGen},D}(k) = \varepsilon'$ en temps τ' tel que

$$\begin{cases} \varepsilon' \geq \varepsilon - \dfrac{q_\Sigma(q_{\mathcal{H}} + q_\Sigma)}{2^{n_r}} \\ \tau' \leq \tau + (q_{\mathcal{H}} + q_\Sigma + O(1))T_{Exp-\mathbb{G}} + (q_\Sigma + O(1))T_{Exp-q} \end{cases}$$

Génération des informations de conversion et vérification :

- TSCUSBM.IReceipt
  • Étant donné un entier $t \in [\![1,T]\!]$, un message m, et une signature (σ, r, t), Alice vérifie sa validité, puis calcule le t-uplet
  $$\tilde{\sigma} = (a_2 H(m||r), a_2^2 H(m||r), \ldots, a_2^t H(m||r))$$
  qui est l'information de vérification de la signature σ. Si elle n'est pas valide, elle donne
  $$\tilde{\sigma} = NIZKP(a_2 : e(\sigma, P_2 + h(m||r)P) \neq e(H(m||r), P_1)^{a_2^t} \wedge P_2 = a_2 P).$$

- TSCUSBM.IVerify
  • Étant donné un entier $t \in [\![1,T]\!]$, un message m et une signature potentielle (σ, r, t) accompagnée de l'information de vérification $\tilde{\sigma} = (\tilde{\sigma}_1, \ldots, \tilde{\sigma}_t)$. Si $\tilde{\sigma} \in \mathbb{G}^t$ et
  $$e(\tilde{\sigma}_1, P) = e(P_2, H(m||r)) \text{ et}$$
  $$e(\tilde{\sigma}_i, P) = e(P_2, \tilde{\sigma}_{i-1}) \quad \forall i \in [\![2,t]\!]$$
  (ce qui signifie que cette information est valide), (σ, r, t) est valide si et seulement si
  $$e(\sigma, P_2 + h(m||r)P) = e(\tilde{\sigma}_t, P_1).$$
  Si $\tilde{\sigma} \notin \mathbb{G}$, il faut vérifier la preuve non-interactive.

- TSCUSBM.TReceipt
  • Étant donné un entier $t \in [\![1,T]\!]$, Alice publie le t-uplet
  $$I_t = (a_1 a_2 P, a_1 a_2^2 P, \ldots, a_1 a_2^t P)$$
  comme reçu de conversion totale.

- TSCUSBM.TVerify
  • Étant donné un entier $t \in [\![1,T]\!]$, le reçu de conversion totale $I_t = (I_1, \ldots, I_t)$ est d'abord vérifiée en s'assurant que
  $$e(P_1, P_2) = e(I_1, P) \text{ et } e(I_{i-1}, P_2) = e(I_i, P) \quad \forall i \in [\![2,t]\!]$$
  • la validité de toute signature (σ, r, t') d'un message m pour une période de temps $t' \leq t$ se vérifie universellement grâce à
  $$e(\sigma, P_2 + h(m||r)P) = e(H(m||r), I_{t'}).$$

FIG. 2.6 – TSCUSBM : description des protocoles de conversion et vérification

où $n_r = f_r(k)$.

PREUVE :

Soit k un paramètre de sécurité, et $(q, P, \mathbb{G}, \mathbb{H}, e)$ un 5-uple généré par $\mathsf{BDHGen}(k)$. Considérons une instance aléatoire (X, Y) du problème CDH.

Jeu_0 On considère un attaquant $\mathsf{EF\text{-}CMA}$ $\mathcal{A}$ de probabilité de succès $\mathbf{Succ}^{\mathsf{ef-cma}}_{\mathsf{TSCUSBM},\mathcal{A}}(k)$ en temps τ. Ce premier jeu est l'attaque réelle menée par $\mathcal{A}$. L'algorithme de génération de clés produit un triplet (pk, sk, T). L'algorithme de génération de reçu de conversion universelle produit le T-uplet $(I_1, \ldots, I_T)$. L'adversaire reçoit la clé publique (pk, T) et le T-uplet $(I_1, \ldots, I_T)$. Ayant interrogé l'oracle aléatoire $\mathcal{H}$ à $q_{\mathcal{H}}$ reprises, l'oracle de signature Σ à q_Σ reprises, et l'oracle de génération de reçu de conversion Υ à q_Υ reprise, il produit un couple message/signature $(m^\star, (\sigma^\star, r^\star, t^\star))$. On demande bien sûr que la signature produite ne soit pas obtenue auprès de l'oracle de signature. Dans les simulations qui vont suivre, il y a au plus $q_{\mathcal{H}} + q_\Sigma + 1$ requêtes à l'oracle aléatoire.
Dans le jeu Jeu_i, on note Forge_i l'événement

$$\mathsf{TSCUSBM.Verify}(k, \mathrm{params}, pk, m^\star, T, (T, t^\star), (\sigma^\star, r^\star, t^\star), I_T) = 1.$$

Par définition,

$$\Pr[\mathsf{Forge}_0] = \mathbf{Succ}^{\mathsf{ef-cma}}_{\mathsf{TSCUSBM},\mathcal{A}}(k)$$

Jeu_1 Dans ce jeu, on tire au hasard un élément $a_2 \in [\![1, q-1]\!]$, puis on modifie la simulation en remplaçant pk par $(X, a_2 P)$ et le T-uplet $(I_1, \ldots, I_T)$ par $(a_2 X, a_2^2 X, \ldots, a_2^T X)$. La distribution de pk est inchangée puisque (X, Y) est une instance aléatoire du problème CDH, et a_2 est tiré au hasard. Le T-uplet $(a_2 X, a_2^2 X, \ldots, a_2^T X)$ est aussi parfaitement simulé. À partir de maintenant, la connaissance de a_2 permet de simuler toute information de conversion, donc Υ. Ainsi,

$$\Pr[\mathsf{Forge}_1] = \Pr[\mathsf{Forge}_0].$$

Jeu_2 Dans ce jeu, on va simuler l'oracle aléatoire $\mathcal{H}$. Pour toute nouvelle requête $(m, r) \in \{0,1\}^* \times \{0,1\}^{n_r}$ à l'oracle $\mathcal{H}$, on tire au hasard un entier $u \in [\![1, q-1]\!]$ et on pose $Q = u(a_2 + h(m||r))Y$. On met à jour la $\mathsf{H\text{-}List}$ en insérant le quadruplet (m, r, u, Q) et on retourne Q comme haché. Dans le modèle de l'oracle aléatoire, ce jeu est identique au jeu précédent :

$$\Pr[\mathsf{Forge}_2] = \Pr[\mathsf{Forge}_1].$$

Jeu_3 Dans ce dernier jeu, on simule l'oracle de signature Σ. Pour tout message $m \in \{0,1\}^*$ dont la signature est demandée pour la période de temps t, on tire au hasard deux éléments $r \in \{0,1\}^{n_r}$ et $v \in [\![1, q-1]\!]$, et on pose $\sigma = a_2^t v (a_2 + h(m||r))^{-1} X$. Si la $\mathsf{H\text{-}List}$ contient un quadruplet $(m, r, ?, ?)$, on arrête la simulation, sinon, on insère dans la $\mathsf{H\text{-}List}$ le quadruplet (m, r, v, vP) et on donne (σ, r, t) comme réponse de l'oracle de signature. Puisque la simulation échoue avec une probabilité au plus $(q_{\mathcal{H}} + q_\Sigma)2^{-n_r}$, on obtient, en faisant la somme de toutes les inégalités pour chaque requête à l'oracle de signature :

$$|\Pr[\mathsf{Forge}_3] - \Pr[\mathsf{Forge}_2]| \leq \frac{q_\Sigma(q_{\mathcal{H}} + q_\Sigma)}{2^{n_r}}.$$

À l'issue de ce dernier jeu, l'attaquant sort une paire message/signature valide $(m^\star, (\sigma^\star, r^\star, t^\star))$, et la H-List contient donc un quadruplet $(m^\star, r^\star, u^\star, Q^\star)$ avec $Q^\star = u^\star(a_2 + h(m||r))Y$. Grâce à cette simulation, on obtient

$$e(\sigma^\star, P_2 + h(m||r)P) = e(Q^\star, a_2^{t^\star} P_1),$$

et donc le point

$$R = \frac{1}{u^\star a_2^{t^\star}} \sigma^\star$$

est une solution à notre instance du problème CDH, ce qui conclut la preuve. □

On en déduit immédiatement la sécurité du schéma de signatures indéniables totalement convertibles CUSBM :

Corollaire 1 (Inforgeabilité de CUSBM) *Sous l'hypothèse CDH, le schéma CUSBM est EF-CMA dans le modèle de l'oracle aléatoire.*

2.4.2 Anonymat

Dans cette section, nous montrons que la propriété d'anonymat des schémas de signatures indéniables précédents repose sur le problème $(q_{\mathcal{H}}, T)$-xyz-DCAA.

Théorème 14 *Soit BDHGen un générateur de groupes BDH symétrique et soit TSCUSBM le schéma de signatures indéniables convertibles dans le temps associé. Soit $f_r : \mathbb{N} \longrightarrow \mathbb{N}$ une fonction. Dans le modèle de l'oracle aléatoire, pour tout adversaire Ano-CMA $\mathcal{A}$ contre TSCUSBM, de paramètre de sécurité k, ayant un avantage pour lever l'anonymat $\mathbf{Adv}^{\mathsf{ano-cma}}_{\mathsf{TSCUSBM},\mathcal{A}}(k) = \varepsilon$ en temps τ, et pouvant faire $q_{\mathcal{H}}$ requêtes a l'oracle aléatoire $\mathcal{H}$, $q_{\hbar}$ à l'oracle aléatoire $\hbar$, q_Σ requêtes à l'oracle de signature et q_Υ requêtes à l'oracle de génération de reçu de conversion, il existe un entier T et un adversaire $\mathcal{D}$ contre $(q_{\mathcal{H}}, T)$-xyz-DCAA, de succès $\mathbf{Adv}^{(\mathsf{q}_{\mathcal{H}},\mathsf{T})\text{-}\mathsf{xyz}\text{-}\mathsf{DCAA}}_{\mathsf{BDHGen},D}(k) = \varepsilon'$ en temps τ' et une fonction négligeable $\nu : \mathbb{N} \longrightarrow \mathbb{R}$ tels que*

$$\begin{cases} \varepsilon' \geq \dfrac{\varepsilon}{2} - \dfrac{(q_\Sigma + 1)(q_{\mathcal{H}} + q_{\hbar})}{2^{n_r - 1}} - \dfrac{1}{2^k} - \nu(k) \\ \tau' \leq \tau + (q_{\mathcal{H}} + 2q_\Sigma + Tq_\Upsilon + O(1))T_{Exp-\mathbb{G}} + (q_\Sigma + Tq_\Upsilon + O(1))T_{Exp-q} \end{cases}$$

où $n_r = f_r(k)$.

Preuve :

Soit k un paramètre de sécurité, T et ℓ deux entiers, et $(q, P, \mathbb{G}, \mathbb{H}, e)$ un 5-uplet généré par BDHGen(k), et $\left[(X_i)_{i \in [\![1, 2T-1]\!]}, Y, Z, h), (R_j, h_j)_{j \in [\![1,\ell]\!]}\right]$ dans $\mathbb{G}^{2T+1} \times [\![1, q-1]\!] \times (\mathbb{G} \times [\![1, q-1]\!])^\ell$ une instance aléatoire du problème (ℓ, T)-xyz-DCAA. On note $X_0 = P$. Nous allons décrire un attaquant $\mathcal{B}$ qui va tenter de résoudre cette instance.

Jeu$_0$ Considérons un attaquant Ano-CMA $\mathcal{A}$ qui brise l'anonymat de TSCUSBM avec un avantage $\mathbf{Adv}^{\mathsf{ano-cma}}_{\mathsf{TSCUSBM},\mathcal{A}}(k)$ en temps τ. Durant ce jeu qui est l'exécution réelle du scénario de l'attaque, l'algorithme de génération de clés est lancé deux fois pour produire les triplets (pk_0, sk_0, T_0) et (pk_1, sk_1, T_1). L'attaquant agit en deux étapes : dans la première, find, il reçoit les clés publiques (pk_0, T_0) et (pk_1, T_1), et accède de façon permanente aux oracles aléatoires $\mathcal{H}$ et $\hbar$, aux oracles de signatures Σ_0 et Σ_1, et aux oracles de génération de reçu de conversion Υ_0 et Υ_1. À l'issue de cette première étape, $\mathcal{A}$ sort un message $m^\star$ et une

période de temps $t^\star \in [\![1, \min(T_0, T_1)]\!]$. Une signature challenge est alors produite de la façon suivante : un bit $b \in \{0,1\}$ est tiré au hasard, puis le message $m^\star$ est signé grâce à la clé secrète sk_b que l'on note $(\sigma^\star, r^\star, t^\star)$. Cette dernière est donnée à l'attaquant qui, à la suite de nouvelles interactions avec les oracles, sort un bit $b^\star$ à la fin de l'étape guess.
On note $q_{\mathcal{H}}$, $q_{\hbar}$, q_Σ et q_Υ le nombre de requêtes à l'oracle aléatoire, à l'oracle de signature, et à l'oracle de génération de reçu de conversion. On suppose $T \geq \min(T_0, T_1)$ et $q_{\mathcal{H}} \leq \ell$. On impose bien sûr que l'attaquant ne demande pas d'information de vérification aux oracles Υ_0 et Υ_1 sur la signature challenge.
Dans chaque jeu Jeu_i, on note Guess_i l'événement $b^\star = b$, et par définition,

$$|2\Pr[\mathsf{Guess}_0] - 1| = \mathbf{Adv}^{\mathsf{ano-cma}}_{\mathsf{TSCUSBM},\mathcal{A}}(k)$$

Jeu_1 Dans ce premier jeu, $\mathcal{B}$ tire au hasard $(\alpha, \beta) \in [\![1, q-1]\!]^2$ et simule la génération des clés en remplaçant pk_0 par $((X_1, Y_1), T_0)$ et pk_1 par $((\alpha X_1, \beta Y_1), T_1)$. Les distributions des clés ne sont pas modifiées puisque l'on se sert d'une instance aléatoire de $(q_{\mathcal{H}}, T)$-xyz-DCAA et d'aléa $(\alpha, \beta) \in [\![1, q-1]\!]^2$. Ainsi,

$$\Pr[\mathsf{Guess}_1] = \Pr[\mathsf{Guess}_0].$$

Jeu_2 Ce sont ici les oracles aléatoires $\mathcal{H}$ et $\hbar$ que $\mathcal{B}$ simule, en maintenant à jour des listes de hachés, que l'on note $\mathsf{H\text{-}List}$ et $\mathsf{h\text{-}List}$, de la façon suivante :
– pour toute requête $(m, r) \in \{0,1\}^*$ à l'oracle $\mathcal{H}$, il tire au hasard $s \in [\![1, q-1]\!]$, calcule $sP \in \mathbb{G}$, stocke dans la $\mathsf{H\text{-}List}$ (m, r, s, sP, T), et retourne sP comme haché de (m, r) ;
– pour toute requête $(m, r) \in \{0,1\}^*$ à l'oracle $\hbar$, il tire au hasard un entier $u \in [\![1, q-1]\!]$, stocke (m, r, u) dans la $\mathsf{h\text{-}List}$ et renvoie u comme réponse de l'oracle.
Dans le modèle de l'oracle aléatoire, ce jeu est identique au précédent :

$$\Pr[\mathsf{Guess}_2] = \Pr[\mathsf{Guess}_1].$$

Jeu_3 La simulation des oracles de signature se fait dans ce troisième jeu : un compteur est initialisé à $i = 1$, et, pour chaque requête de signature du message $m \in \{0,1\}^*$ à la période $t \in [\![1, T]\!]$, $\mathcal{B}$ tire un entier $r \in \{0,1\}^{n_r}$ au hasard. S'il existe un triplet $(m, r, ?)$ dans la $\mathsf{h\text{-}List}$ ou un 5-uplet $(m, r, ?, ?, ?)$ dans la $\mathsf{H\text{-}List}$, $\mathcal{B}$ s'arrête. Sinon, il tire au hasard $s \in [\![1, q-1]\!]$, calcule $sX_{T-t} \in \mathbb{G}$ et stocke dans la $\mathsf{H\text{-}List}$ (m, r, s, sX_{T-t}, t).
Si la requête s'adresse à Σ_0, la signature est (sR_i, r, t), et $\mathcal{B}$ met à jour la $\mathsf{h\text{-}List}$ avec (m, r, h_i), puis incrémente le compteur. Si la requête s'adresse à Σ_1, la signature est $(s\alpha^{t-1}\beta R_i, r, t)$, et $\mathcal{B}$ met à jour la $\mathsf{h\text{-}List}$ avec $(m, r, \alpha h_i)$, puis incrémente le compteur.
La simulation des oracles dans ce jeu est parfaite si la simulation ne s'arrête pas, ce qui arrive avec une probabilité au plus $(q_{\mathcal{H}} + q_{\hbar})2^{-n_r}$ pour chaque requête aux oracles de signature, et donc :

$$|\Pr[\mathsf{Guess}_3] - \Pr[\mathsf{Guess}_2]| \leq \frac{q_\Sigma(q_{\mathcal{H}} + q_{\hbar})}{2^{n_r}}.$$

Jeu_4 Dans ce jeu, $\mathcal{B}$ va simuler les deux oracles de génération de reçu de conversion. Si l'adversaire s'adresse à Υ_0 avec un couple message/signature potentielle $(m, (\sigma, r, t))$, $t \in [\![1, T]\!]$, $\mathcal{B}$ cherche dans la $\mathsf{H\text{-}List}$ un 5-uplet $(m, r, s, sX_{T-t'}, t')$. Si la signature est valide, la réponse donnée par $\mathcal{B}$ est le t-uplet $(sX_{T-t'+1}, \ldots, sX_{T-t'+t})$. Si la requête s'adresse à Υ_1, la réponse renvoyée si la signature est valide est $(s\alpha^{T-t'+1}X_{T-t'+1}, \ldots, s\alpha^{T-t'+t}X_{T-t'+t})$. Dans le cas où la signature est invalide, il faut simuler la preuve non-interactive d'inégalité de logarithmes discrets, ce qui est possible dans le modèle de l'oracle aléatoire, et qui peut introduire un biais négligeable dans la simulation. Il existe donc une fonction négligeable $\nu : \mathbb{N} \longrightarrow \mathbb{R}$ telle que :

$$|\Pr[\mathsf{Guess}_4] - \Pr[\mathsf{Guess}_3]| \leq \nu(k).$$

Jeu_5 Lorsque l'adversaire $\mathcal{A}$ demande à son challenger une signature sur le message challenge $m^\star$ qu'il a choisi, pour une période de temps $t^\star$, $\mathcal{B}$ s'adresse à l'oracle associé au problème (ℓ, T)-xyz-DCAA en lui soumettant $t^\star$. L'oracle renvoie à $\mathcal{B}$ le $(t^\star - 1)$-uplet $(Y_j)_{j \in [\![1, t^\star - 1]\!]} \in \mathbb{G}^{t^\star - 1}$ et $Q \in \mathbb{G}$. $\mathcal{B}$ tire alors un bit $b \in \{0, 1\}$ et $r^\star \in \{0, 1\}^{n_r}$ au hasard. S'il y a un 5-uplet $(m^\star, r^\star, ?, ?, ?)$ dans la $\mathsf{H\text{-}List}$ ou un triplet $(m^\star, r^\star, ?)$ dans la $\mathsf{h\text{-}List}$, $\mathcal{B}$ s'arrête, sinon, il met à jour la $\mathsf{H\text{-}List}$ en y insérant $(m^\star, r^\star, \perp, Z, \perp)$ et la $\mathsf{h\text{-}List}$ en y insérant $(m^\star, r^\star, \alpha^b, h)$. $\mathcal{B}$ propose à $\mathcal{A}$ la signature challenge $((\alpha^{t^\star - 1}\beta)^b Q, r^\star, t^\star)$ et $(Y_j)_{j \in [\![1, t^\star - 1]\!]} \in \mathbb{G}^{t^\star - 1}$ comme reçus universels pour le challenge jusqu'à la période $t^\star - 1$.
Si $Q = Q_{\mathsf{real}} = x^T yz(x + h)^{-1} P$, ce jeu simule parfaitement la génération de la signature challenge, s'il ne se termine pas prématurément, ce qui arrive avec une probabilité au plus $(q_{\mathcal{H}} + q_{\hbar}) 2^{-n_r}$. Ainsi,

$$|\Pr[\mathsf{Guess}_5 | Q = Q_{\mathsf{real}}] - \Pr[\mathsf{Guess}_4]| \leq \frac{q_{\mathcal{H}} + q_{\hbar}}{2^{n_r}}$$

Si $Q = Q_{\mathsf{random}}$ est un élément quelconque de $\mathbb{G}$, l'adversaire ne gagne absolument aucune information sur b et donc

$$\Pr[\mathsf{Guess}_5 | Q = Q_{\mathsf{random}}] \leq \frac{1}{2} + \frac{q_{\mathcal{H}} + q_{\hbar}}{2^{n_r}} + \frac{1}{2^k}.$$

Le dernier terme mesure la probabilité que $Q_{\mathsf{random}} = Q_{\mathsf{real}}$. Par définition, l'avantage dans le jeu Jeu_5 de $\mathcal{B}$ pour résoudre le problème $(q_{\mathcal{H}}, T) - xyz$-DCAA est :

$$\mathbf{Adv}^{(q_{\mathcal{H}}, \mathsf{T}) - \mathsf{xyz} - \mathsf{dcaa}}_{\mathcal{G}en, \mathsf{Game}_5}(k) = \big| \Pr[\mathsf{Guess}_5 | Q = Q_{\mathsf{real}}] - \Pr[\mathsf{Guess}_5 | Q = Q_{\mathsf{random}}] \big|.$$

Ceci termine la preuve.

□

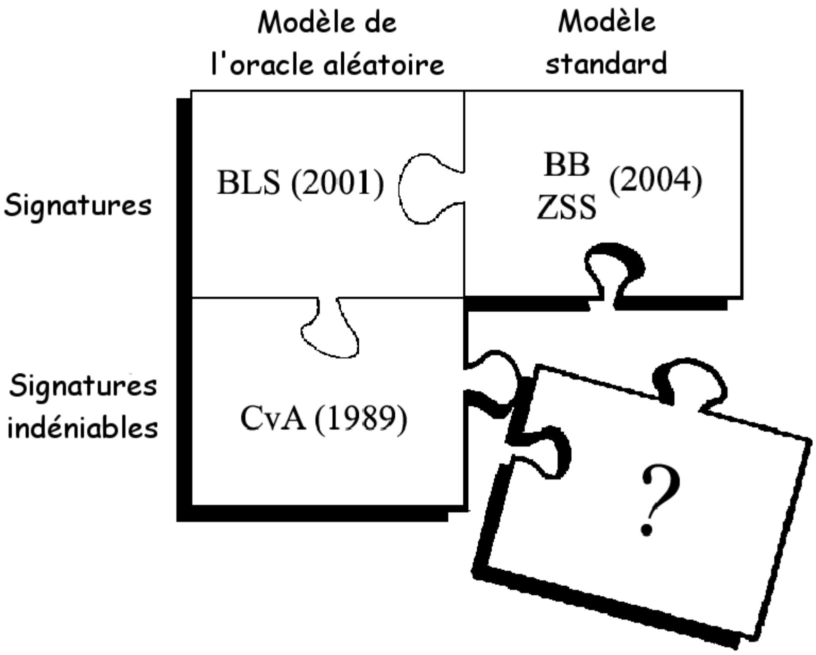
L'anonymat du schéma de signatures indéniables totalement convertibles CUSBM découle du théorème précédent :

Corollaire 2 *Sous l'hypothèse $(\ell, 1)$-xyz-DCAA, le schéma CUSBM est $\mathsf{Ano\text{-}CMA}$ dans le modèle de l'oracle aléatoire.*

2.5 Signatures indéniables courtes dans le modèle standard

Dans cette section, nous proposons un schéma de signatures indéniables dont la sécurité est prouvée dans le modèle standard. C'est une translation du schéma de Boneh et Boyen [34] vers des groupes où le problème Diffie-Hellman décisionnel est supposé difficile (par exemple sur une courbe elliptique de grand degré MOV). Nous utilisons une technique de preuve similaire à celle de Boneh et Boyen : la présence d'une deuxième clé secrète (qui peut se voir comme la trappe d'une fonction de hachage caméléon) et d'une hypothèse particulière (q-SDH) donne suffisamment de liberté pour se passer de l'hypothèse idéale d'oracle aléatoire.

De la même façon que l'existence des applications bilinéaires admissibles permet de rendre "ordinaires" les signatures de Chaum et Van Antwerpen [57], l'inexistence de celles-ci va rendre les signatures de Boneh-Boyen [34] non universellement vérifiables. Ainsi, grâce à des protocoles zero-knowledge interactifs d'appartenance à un langage, la vérification va être rendue possible par une intervention du signataire.



Ainsi, le schéma proposé ici est au schéma de signatures de Boneh et Boyen ce que le schéma de Chaum et van Antwerpen est au schéma BLS, et il est au schéma de Chaum et van Antwerpen ce que celui de Boneh et Boyen est à BLS.

2.5.1 Preuve d'égalité de logarithme discret

Dans notre schéma, les protocoles de confirmation et de démenti sont obtenus grâce à des preuves interactives zero-knowledge non-transférables. Ce sont en fait des versions à vérificateur désigné des preuves d'égalité (voir la section 1.2.5) et d'inégalité de logarithmes discrets de Chaum [54]. Ces preuves sont décrites dans les figures 2.7 et 2.8. Le vérificateur possède une paire de clés $(y, P_{\mathcal{V}})$, où $P_{\mathcal{V}} = yP$.

***Remarque* 6** *Dans [126], Jakobsson et Schnorr proposent un protocole d'appartenance au langage des quadruplets Diffie-Hellman valides, ce qui leur permet de n'avoir qu'un protocole pour la confirmation et le démenti, et plus efficace.*

Un quadruplet (P, Q, R, S) est dans le langage des quadruplets Diffie-Hellman valide si et seulement si $\log_P(Q) = \log_R(S)$. Le protocole fonctionne de la façon suivante : le prouveur connaît $x = \log_P(Q)$ et veut donc déterminer et prouver que $\log_P(Q) = \log_R(S)$.

1. Setup *: le prouveur tire $a \in [\![1, q-1]\!]$ au hasard.*
2. Preuve de premier ordre *: le prouveur génère un triplet $(\overline{S}, \overline{\sigma}, \overline{R}) = (aS, axR, aR)$. Un vérificateur accepte cette première preuve si et seulement si $\overline{S} = \overline{\sigma}$.*
3. Preuve de second ordre *: le prouveur prouve que $\log_R(\overline{R}) = \log_S(\overline{S})$ et que $\log_P(Q) = \log_{\overline{R}}(\overline{\sigma})$ avec la preuve zero knowledge d'égalité de logarithmes discrets décrite dans la figure 1.1. Le vérificateur accepte cette seconde preuve si et seulement si les deux équations sont vérifiées.*

Prouveur		Vérificateur
		$(a,b) \in_R [\![1, q-1]\!]^2$
	$\xleftarrow{\quad C \quad}$	$C = aL + bN$
$r \in_R [\![1, q-1]\!]$		
$R_1 = C + rP_{\mathcal{V}}$		
$R_2 = xR_1$	$\xrightarrow{\quad R_1, R_2 \quad}$	
	$\xleftarrow{\quad a, b \quad}$	
$C \stackrel{?}{=} aL + bN$	$\xrightarrow{\quad r \quad}$	
		$R_1 \stackrel{?}{=} C + rP_{\mathcal{V}}$
		$R_2 \stackrel{?}{=} (a + yr)M + bO$

FIG. 2.7 – Preuve non transférable de l'égalité $\log_L(M) = \log_N(O) = x$

Prouveur		Vérificateur
		$s \in_R [\![0, \lambda]\!]$ et $a \in_R [\![1, q-1]\!]$
	$\xleftarrow{\quad C, C' \quad}$	$C = aL + sN$ et $C' = aM + sN$
cherche s' tel que $(xC - C') = s'(xN - O)$		
$r \in_R [\![1, q-1]\!]$		
$C'' = s'P + rP_{\mathcal{V}}$	$\xrightarrow{\quad C'' \quad}$	
	$\xleftarrow{\quad a \quad}$	
$C \stackrel{?}{=} aL + s'N$	$\xrightarrow{\quad (s', r) \quad}$	$R \stackrel{?}{=} s'P + rP_{\mathcal{V}}$ et $s \stackrel{?}{=} s'$

FIG. 2.8 – Preuve non transférable de l'inégalité $\log_L(M) \neq \log_N(O)$

4. Décision *: le vérificateur déclare* `valid` *si les deux preuves sont correctes. Il déclare* `invalid` *s'il a rejeté la preuve de premier ordre et accepté la preuve de second ordre. Sinon il est en face d'un prouveur qui triche (*`cheating prover !`*).*

Dans [126], Jakobsson et Schnorr démontrent que ce protocole est consistant, significatif, et "inconscient", c'est-à-dire que le prouveur exécute le même protocole, que les quadruplets d'entrée soient dans le langage ou non. De plus, ils montrent le résultat suivant :

Proposition 4 *Si l'hypothèse Diffie-Hellman décisionelle est vraie, alors le protocole précédent pour décider une exponentiation correcte est* calculatoirement à minimum de connaissance *(i.e étant donné un oracle pour l'appartenance au langage, il est possible de simuler en temps polynomial une vue du protocole qui ne pourra pas être distinguée avec une probabilité non-négligeable de celle d'un vrai protocole pour la preuve correspondante).*

2.5.2 USWRO : description

La figure 2.9 décrit le nouveau schéma de signatures indéniables USWRO. La génération d'une signature coûte essentiellement une multiplication scalaire sur la courbe. La signature consiste alors en un point d'une courbe elliptique et d'un aléa. Les protocoles de confirmation et de démenti sont des preuves classiques d'égalité et d'inégalité de logarithmes discrets : la preuve d'inégalité est toutefois relativement peu efficace puisqu'il y a une petite recherche exhaustive à faire.

2.5.3 Preuves de sécurité

Dans cette section, nous analysons la sécurité de USWRO dans le modèle standard. Nous montrons qu'il reste anonyme contre un adversaire Ano-CMA dans une attaque à messages choisis si DSDH est difficile et si USWRO résiste à un attaquant EF-CMA dans une attaque à messages choisis. Nous montrons ensuite qu'il résiste effectivement à un attaquant EF-CMA dans une attaque à messages choisis si le problème GSDH est difficile.

2.5.3.1 Anonymat

Pour tout adversaire Ano-CMA $\mathcal{A}$, on note $\mathsf{Bad}_{\mathcal{A}}$ l'évènement qui consiste pour $\mathcal{A}$ à demander à un oracle de confirmation ou de démenti la vérification d'une signature valide qui n'a pas été obtenue auprès d'un oracle de signature Σ. Nous allons alors décrire deux algorithmes $\mathcal{C}$ et $\mathcal{D}$ qui vont résoudre ℓ-DSDH ou forger une signature indéniable, selon que $\Pr[\mathsf{Bad}]$ est négligeable ou pas.

Proposition 5 *Soit* GGen *un générateur de groupes d'ordre premier et* USWRO *le schéma de signatures indéniables associé. Soit $\mathcal{A}$ un adversaire* Ano-CMA *contre* USWRO *de paramètre de sécurité k et d'avantage $\mathbf{Adv}^{\mathsf{ano-cma}}_{\mathsf{USWRO},\mathcal{A}}(k)$ en temps τ faisant q_Σ requêtes aux oracles de signature, q_C aux oracles de confirmation et q_D aux oracles de démenti. Il peut aussi enregistrer jusqu'à $q_\mathcal{R}$ clés. Alors il existe un adversaire $\mathcal{B}$ contre $(q_\Sigma+1)$-DSDH d'avantage $\mathbf{Adv}^{(q_\Sigma+1)\mathsf{dsdh}}_{\mathsf{GGen},\mathcal{B}}(k) = \varepsilon'$ en temps τ' tel que*

$$\begin{cases} \varepsilon' \geq \dfrac{\varepsilon}{2} - \dfrac{q_\Sigma + 2}{2^k} - \Pr[\mathit{Bad}] \\ \tau' \leq (q_C + q_\mathcal{R})\tau + q_\Sigma(q_\Sigma T_\mathbb{G} + O(1)). \end{cases}$$

Génération des paramètres publics : $\mathsf{USWRO.Setup}$

- k est un paramètre de sécurité,
- $(q, P, \mathbb{G})$ est généré par $\mathsf{GGen}(k)$

Génération des clés : $\mathsf{USWRO.KeyGen}$

- Alice tire au hasard $(a_1, a_2) \in [\![1, q-1]\!]^2$ qui est la clé secrète.
- Sa clé publique est (P_1, P_2), avec $P_1 = a_1 P$ et $P_2 = a_2 P$.

Signature : $\mathsf{USWRO.Sign}$

- Étant donné un message m, et les clés secrètes de signature a_1 et a_2, Alice tire au hasard un élément $r \in [\![1, q-1]\!]$ et calcule $t = a_1 + m + ra_2$. Si $t = 0$, elle tire un nouveau r, et sinon, elle calcule le point

$$S = \frac{1}{a_1 + m + ra_2} P.$$

- La signature du message m est $\sigma = (S, r)$.

Protocoles de confirmation et de démenti : $\mathsf{USWRO.\{Confirm,Deny\}}$

- Étant donné un message m et une signature $\sigma = (S, r)$, Alice confirme ou dément celle-ci grâce à la preuve interactive de connaissance suivante :

$$IZKP(\log_S(P - mS) = \log_P(P_1 + rP_2)) \text{ ou}$$

$$IZKP(\log_S(P - mS) \neq \log_P(P_1 + rP_2))$$

FIG. 2.9 – USWRO : description

PREUVE :

Soit $k \in \mathbb{N}$ un paramètre de sécurité, GGen un générateur de groupes d'ordre premier et $(q, \mathbb{G}) = \mathsf{GGen}(k)$. Le $\ell + 3$-uplet $(P, xP, x^2P, \ldots, x^\ell P, X, h) \in \mathbb{G}^{\ell+2} \times [\![1, q-1]\!]$ est une instance aléatoire de ℓ-DSDH. On peut supposer que $q_\Sigma = \ell + 1$. Notons A_i le point $x^i P$ pour $i \in [\![1, \ell]\!]$.

Jeu_0 Soit $\mathcal{A}$ un adversaire Ano-CMA contre USWRO, ayant un avantage $\mathbf{Adv}^{\mathsf{ano-cma}}_{\mathsf{USWRO},\mathcal{A}}(k)$ au bout d'un temps τ. L'algorithme de génération de clés est lancé deux fois pour produire deux paires de clés (pk_0, sk_0) et (pk_1, sk_1). Dans sa première étape Find, $\mathcal{A}$ reçoit les clés publiques pk_0 et pk_1 et entreprend d'interroger les oracles de signature Σ_0 et Σ_1, les oracles de confirmation Υ_C^0 et Υ_C^1 et les oracles de démenti Υ_D^0 et Υ_D^1 jusqu'au moment où il sort un message $m^\star$. Son challenger tire un bit $b^\star \in \{0, 1\}$ au hasard et obtient auprès de l'oracle de signature $\Sigma_{b^\star}$ une signature du message $m^\star$, notée $\sigma^\star$ qu'il renvoie à l'attaquant. Commence alors la deuxième phase de l'attaquant qui, une fois en possession de $\sigma^\star$, a à nouveau un accès permanent aux différents oracles, avec la restriction de ne pas interroger les oracles

de confirmation et de démenti à propos de la signature challenge.
On note q_Σ le nombre de requêtes aux oracles de signature et q_C et q_D les nombres de requêtes aux oracles de confirmation et de démenti.

Dans toute la suite de la démonstration, pour tout jeu Jeu_j, on note Guess_j l'événement $b = b^\star$. Par définition,

$$|2\Pr[\mathsf{Guess}_0] - 1| = \mathbf{Adv}^{\mathsf{ano-cma}}_{\mathsf{USWRO},\mathcal{A}}(k)$$

Jeu_1 L'algorithme $\mathcal{B}$ commence par tirer aléatoirement (u, v, a_0, a_1) dans $[\![1, q-1]\!]^4$, puis initialise un compteur c à 1, et deux listes Σ-List et $\overline{\Sigma}$-List à $\{\}$. Ensuite, il tire aléatoirement $h_i \in [\![1, q-1]\!]$ pour $i \in [\![1, q_\Sigma]\!]$. Si $h \in \{h_1, \ldots, h_{q_\Sigma}\}$ ou $uh \in \{h_1, \ldots, h_{q_\Sigma}\}$, alors $\mathcal{B}$ s'arrête, ce qui arrive avec une probabilité au plus $q_\Sigma 2^{1-k}$. $\mathcal{B}$ prépare alors le polynôme suivant :

$$f(y) = \prod_{i=1}^{q_\Sigma}(y + h_i) = \sum_{i=0}^{q_\Sigma} \alpha_i y^i \in \mathbb{F}_q[y],$$

et les points

$$P' = \sum_{i=0}^{q_\Sigma} \alpha_i A_i = f(x)P,$$

$$P_0 = \sum_{i=1}^{q_\Sigma+1} \alpha_{i-1} A_i = xf(x)P = xP' \text{ et } P_1 = uvP_0 = uvxP'$$

Il pose enfin $pk_0 = (P', P_0, a_0 P')$ et $pk_1 = (vP', P_1, a_1 vP')$, qu'il donne à $\mathcal{A}$ comme les clés publiques des deux signataires potentiels. La distribution de (pk_0, pk_1) reste inchangée, puisqu'on considère une instance aléatoire de ℓ-DSDH et que (u, v, a_0, a_1) sont tirés aléatoirement. Aussi

$$\Pr[\mathsf{Guess}_1] = \Pr[\mathsf{Guess}_0].$$

Jeu_2 À partir de cette étape, dès que $\mathcal{A}$ enregistre une clé publique destinée à être utilisée dans les protocoles de confirmation et de démenti, $\mathcal{B}$ extrait la clé secrète correspondante en rembobinant la simulation, et en la relançant avec les mêmes bandes aléatoires, jusqu'au moment de cet enregistrement. Il biaise alors sa réponse pour extraire la clé secrète. Maintenant, $\mathcal{B}$ connaît les clés secrètes de l'attaquant, et la simulation est parfaite :

$$\Pr[\mathsf{Guess}_2] = \Pr[\mathsf{Guess}_1].$$

Jeu_3 Dans ce jeu, $\mathcal{B}$ simule les oracles de signatures. Il initialise un compteur à $c = 1$ et pour toute nouvelle requête sur $m \in \{0,1\}^*$, $\mathcal{B}$ construit le polynôme

$$f_c(y) = \frac{f(y)}{y + h_c} = \prod_{\substack{j=1 \\ j \neq c}}^{q_\Sigma} (y + h_j) = \sum_{j=0}^{q_\Sigma - 1} \beta_j^{(c)} y^j,$$

et calcule

$$S_c = \frac{v^b}{u^b} \sum_{j=0}^{q_\Sigma - 1} \beta_j^{(c)} A_j = \frac{v^b}{u^b} \sum_{j=0}^{q_\Sigma - 1} \beta_j^{(c)} x^j P = f_c(x)P = \frac{v^b}{u^b} \frac{f(x)}{x + h_c} P = \frac{v^b}{u^b(x + h_c)} P',$$

où $b \in \{0,1\}$ est déterminé par l'oracle de signature Σ_b interrogé. $\mathcal{B}$ pose $r_c = (h_c u^b - m_c) a_2^{-1}$ et si $r_c = 0$, alors $\mathcal{B}$ s'arrête, sinon, il ajoute $(m_c, (S_c, r_c), b)$ à la Σ-List, sort (S_c, r_c) comme signature valide de m_c par rapport à la clé publique pk_b et incrémente son compteur. Durant son exécution, $\mathcal{B}$ s'arrête au cours de la simulation des signatures avec une probabilité au plus $q_\Sigma 2^{-k}$. Aussi

$$|\Pr[\mathsf{Guess}_3] - \Pr[\mathsf{Guess}_2]| \leq \frac{q_\Sigma}{2^k}.$$

Jeu_4 Lorsqu'une signature potentielle (S, r) sur un message m est proposée à l'oracle de confirmation Υ_C^b, avec $b \in \{0,1\}$, $\mathcal{B}$ vérifie tout d'abord que $(m, (S,r), b)$ appartient bien à la Σ-List. Si ce n'est pas le cas, il l'ajoute à la $\overline{\Sigma}$-List, et s'arrête. Dans l'autre cas, il pose $R = v^b P' - mS$ et $Q = P_b + r a_b v^b P'$ et il simule la preuve de l'égalité de $\log_S(R)$ et $\log_{v^b P'}(Q)$ de la façon suivante : lorsque $\mathcal{A}$ lui envoie son engagement C, $\mathcal{B}$ tire au hasard $\gamma \in [\![1, q-1]\!]$ et calcule $R_1 = \gamma v^b P'$ et $R_2 = \gamma Q$ et envoie (R_1, R_2) à $\mathcal{A}$. Ce dernier ouvre son engagement (a, b), et $\mathcal{B}$ vérifie que $C = av^b P' + bR$. Si ce n'est pas le cas $\mathcal{A}$ n'est pas honnête. En cas d'égalité, donc, $\mathcal{B}$ rembobine $\mathcal{A}$, et le réinitialise avec la même bande aléatoire puis rejoue exactement la même simulation jusqu'au moment où $\mathcal{A}$ lui envoie C. Maintenant qu'il connaît (a, b) et y (obtenu lors de l'enregistrement), $\mathcal{B}$ peut calculer, après avoir tiré $r \in [\![1, q-1]\!]$ au hasard, $R_1 = C + rP_{\mathcal{A}}$ (où $P_{\mathcal{A}} = yv^b P'$) et $R_2 = (a + yr)Q + bS$, qu'il envoie à $\mathcal{A}$, qui accepte la preuve.

Losque $\mathcal{A}$ fait une requête à l'oracle de démenti Υ_D^b, avec $b \in \{0,1\}$, sur une signature potentielle (S, r) de m, $\mathcal{B}$ vérifie que σ n'apparaît pas dans la Σ-List (si c'est le cas, il indique qu'elle est valide) et rajoute (S, r) à la $\overline{\Sigma}$-List. $\mathcal{B}$ simule alors le protocole de démenti de la façon suivante : il calcule $R = v^b P' - mS$ et $Q = P_b + r a_b v^b P'$ puis quand $\mathcal{A}$ lui envoie (C, C') comme engagement, $\mathcal{B}$ tire au hasard $\tilde{r} \in [\![0, \lambda]\!]$ et envoie à $\mathcal{A}$ $C'' = \tilde{r} v^b P'$. $\mathcal{A}$ lui répond par a, et $\mathcal{B}$ cherche alors $s' \in [\![0, \lambda]\!]$ tel que $C = av^b P' + s'R$. S'il n'en trouve pas, il s'arrête ($\mathcal{A}$ ne joue pas le jeu). Sinon $\mathcal{B}$ calcule $r = (\tilde{r} - s')y^{-1}$ de telle sorte que $C'' = s'P + rP_{\mathcal{A}}$ (où $P_{\mathcal{A}} = yv^b P'$). Finalement, $\mathcal{B}$ renvoie (s', r). Cette simulation est parfaite :

$$\Pr[\mathsf{Guess}_4] = \Pr[\mathsf{Guess}_3].$$

Jeu_5 $\mathcal{B}$ simule maintenant la signature challenge. La division euclidienne de $f(y)$ par $(y+h)$ donne

$$\frac{f(y)}{y+h} = \frac{\gamma}{x+h} + \sum_{i=0}^{q_\Sigma - 2} \gamma_i y^i.$$

$\mathcal{B}$ tire alors $b^\star \in \{0,1\}$ au hasard et la signature challenge simulée est $(S^\star, r^\star)$ avec

$$S^\star = \frac{v^{b^\star}}{u^{b^\star}} \gamma X + \sum_{i=1}^{q_\Sigma - 2} \gamma_i (uv)^{b^\star (i-1)} A_i \text{ et } r^\star = (u^{b^\star} h - m^\star) a_2^{-1}$$

qui est non nul avec une probabilité au plus 2^{-k}. Si cela arrive, $\mathcal{B}$ s'arrête, sinon, il donne $(S^\star, r^\star)$ comme signature challenge à $\mathcal{A}$. À moins que $\mathcal{B}$ s'arrête, cette simulation est parfaite.

Cela termine la description de $\mathcal{B}$. Si $Q = Q_{\mathsf{real}} = (x+h)^{-1} P$, ce jeu simule parfaitement la génération de la signature challenge si l'évènement Bad ne se produit pas et si $\mathcal{B}$ ne s'arrête pas (ce qui arrive avec une probabilité au plus 2^{-k}). Aussi

$$|\Pr[\mathsf{Guess}_5 | Q = Q_{\mathsf{real}}] - \Pr[\mathsf{Guess}_4]| \leq \frac{1}{2^k} + \Pr[\mathsf{Bad}].$$

Si $Q = Q_{\mathsf{random}}$ est un élément aléatoire de $\mathbb{G}$, l'adversaire n'obtient aucune information sur b et donc

$$\Pr[\mathsf{Guess}_5 | Q = Q_{\mathsf{random}}] \leq \frac{1}{2} + \frac{1}{2^k} + \frac{1}{2^k}.$$

Le dernier terme désigne la probabilité que $Q_{\mathsf{random}} = Q_{\mathsf{real}}$. Par définition,

$$\mathbf{Adv}^{(q_\Sigma+1)-\mathsf{dsdh}}_{\mathsf{GGen},\mathsf{Game}_5}(k) = |\Pr[\mathsf{Guess}_5|Q = Q_{\mathsf{real}}] - \Pr[\mathsf{Guess}_5|Q = Q_{\mathsf{random}}]|\,.$$

Cela implique les bornes annoncées.

□

Proposition 6 *Soit GGen un générateur de groupes d'ordre premier et USWRO le schéma de signatures indéniables associé. Soit $\mathcal{A}$ un adversaire Ano-CMA contre USWRO de paramètre de sécurité k et d'avantage $\mathbf{Adv}^{\mathsf{ano-cma}}_{\mathsf{USWRO},\mathcal{A}}(k)$ en temps τ faisant q_Σ requêtes aux oracles de signature, q_C aux oracles de confirmation et q_D aux oracles de démenti. Il peut aussi enregistrer jusqu'à $q_{\mathcal{R}}$ clés. Alors il existe un adversaire EF-CMA $\mathcal{C}$ contre USWRO de succès $\varepsilon'' = \mathbf{Succ}^{\mathsf{ef-cma}}_{\mathsf{USWRO},\mathcal{A}}(k)$, et temps τ'', faisant q_Σ requêtes à l'oracle de signature, q_C à l'oracle de confirmation, q_D à celui de démenti et enregistre jusqu'à $q_{\mathcal{R}}$ clés et tel que*

$$\left\{\ \varepsilon'' \geq \Pr[\mathit{Bad}_{\mathcal{A}}] \text{ and } \tau'' \leq \tau + O(1).\right.$$

Preuve :

Soit k un paramètre de sécurité, GGen un générateur de groupes d'ordre premier et $(q, \mathbb{G}) = \mathsf{GGen}(k)$. On considère une clé publique aléatoire pk et nous allons construire une machine qui va produire une forge existentielle associée à pk.

Jeu_0 Ce premier jeu est exactement le même que dans la preuve de la proposition précédente. Par définition :

$$|2\Pr[\mathsf{Guess}_0] - 1| = \mathbf{Adv}^{\mathsf{ano-cma}}_{\mathsf{USWRO},\mathcal{A}}(k).$$

Jeu_1 Dans ce jeu, l'algorithme $\mathcal{C}$ intéragit avec $\mathcal{A}$ de la façon suivante : lorsque son challenger lui donne une clé publique pk, $\mathcal{C}$ transmet à $\mathcal{A}$ $pk_0 = pk$ et pk_1 construite comme dans la simulation de $\mathcal{B}$. simule les oracles de signature, confirmation et démenti de $\mathcal{A}$ grâce à ses propres oracles. À la fin de son étape `Find`, $\mathcal{A}$ soumet à son challenger un message $m^\star$. $\mathcal{C}$ va simuler ce challenger en tirant un bit $b^\star$ au hasard et, si $b^\star = 1$ il demande à son oracle de signature une signature de $m^\star$, et si $b^\star = 0$, il tire un élément au hasard dans $\mathbb{G} \times [\![1, q-1]\!]$. On note $\sigma^\star$ l'élément qu'il a obtenu. Il le renvoie à $\mathcal{A}$ qui se lance dans son étape `Guess`. La simulation des oracles de $\mathcal{A}$ se fait comme précédemment grâce aux oracles de $\mathcal{C}$. À la fin de cette dernière étape, $\mathcal{A}$ sort un bit. La valeur de ce bit n'a pas d'importance pour $\mathcal{C}$. Durant ce jeu, $\mathcal{C}$ a classé toute les paires message/signature acceptées par l'oracle de confirmation, pour lesquelles la signature n'a pas été obtenue auprès de son oracle de signature. Finalement, $\mathcal{C}$ sort un élément quelconque de la liste, si elle n'est pas vide, et un élément aléatoire de $\{0,1\}^* \times \mathbb{G} \times [\![1, q-1]\!]$ sinon. Ce jeu est identique au précédent :

$$\Pr[\mathsf{Guess}_1] = \Pr[\mathsf{Guess}_0].$$

Son temps d'exécution est celui de $\mathcal{A}$, et son succès au moins $\Pr[\mathsf{Bad}]$.

□

On peut alors énoncer le théorème suivant, qui est une conséquence des deux propositions précédentes :

Théorème 15 (Anonymat de USWRO) *Soit GGen un générateur de groupes d'ordre premier et USWRO le schéma de signatures indéniables associé. Soit $\mathcal{A}$ un adversaire Ano-CMA contre USWRO de paramètre de sécurité k et d'avantage $\mathbf{Adv}^{\mathsf{ano-cma}}_{\mathsf{USWRO},\mathcal{A}}(k)$ en temps τ faisant q_Σ requêtes aux oracles de signature, q_C aux oracles de confirmation et q_D aux oracles de démenti. Il peut aussi enregistrer jusqu'à $q_{\mathcal{R}}$ clés. Alors il existe*

- *un adversaire* $\mathcal{B}$ *contre* $(q_\Sigma+1)$*-DSDH d'avantage* $\mathbf{Adv}^{(q_\Sigma+1)\mathsf{dsdh}}_{\mathsf{GGen},\mathcal{B}}(k)=\varepsilon'$ *en temps* τ' *;*
- *un adversaire EF-CMA* $\mathcal{C}$ *contre USWRO de succès* $\varepsilon''=\mathbf{Succ}^{\mathsf{ef-cma}}_{\mathsf{USWRO},\mathcal{A}}(k)$*, et temps* τ''*, faisant* q_Σ *requêtes à l'oracle de signature,* q_C *à l'oracle de confirmation,* q_D *à celui de démenti et enregistre jusqu'à* $q_\mathcal{R}$

tels que

$$\begin{cases} \varepsilon'+\varepsilon''\geq \dfrac{\varepsilon}{2}-\dfrac{q_\Sigma+2}{2^k} \\ \tau'\leq (q_C+q_\mathcal{R})\tau+q_\Sigma(q_\Sigma T_\mathbb{G}+O(1)) \\ \tau''\leq \tau+O(1). \end{cases}$$

2.5.3.2 Inforgeabilité

Théorème 16 (Inforgeabilité de USWRO) *Soit GGen un générateur de groupes d'ordre premier et USWRO le schéma de signatures indéniables associé. Soit* $\mathcal{A}$ *un adversaire EF-CMA contre USWRO de succès* $\mathbf{Succ}^{\mathsf{ef-cma}}_{\mathsf{USWRO},\mathcal{A}}(k)=\varepsilon$ *en temps* τ*, faisant* q_Σ *requêtes à l'oracle de signature,* q_Υ *à l'oracle de confirmation et démenti. Alors il existe un adversaire* $\mathcal{B}$ *contre* $(q_\Sigma+1)$*-GSDH d'avantage* $\mathbf{Adv}^{(q_\Sigma+1)\mathsf{gsdh}}_{\mathsf{GGen},\mathcal{B}}(k)=\varepsilon'$ *en temps* τ' *tel que*

$$\begin{cases} \varepsilon'\geq \dfrac{\varepsilon}{2}-\dfrac{q_\Sigma}{2^k} \\ \tau'\leq q_R\tau+(q_C+q_D)T_{DSDH}+q_\Sigma O(1). \end{cases}$$

Preuve : (esquisse)

Soit k un paramètre de sécurité, et $\mathcal{A}$ un attaquant EF-CMA de succès $\mathbf{Succ}^{\mathsf{ef-cma}}_{\mathsf{USWRO},\mathcal{A}}(k)$ en temps τ. L'agorithme de génération de clés produit une paire de clés (pk, sk). $\mathcal{A}$ reçoit pk et interroge les oracles de signature Σ, de conversion Υ_C et de démenti Υ_D respectivement à q_Σ, q_C et q_D reprises. Puis il renvoie un couple message/signature $(m^\star,\sigma^\star)$. La signature produite $\sigma^\star$ n'a pas été obtenue auprès de l'oracle de signature. La preuve est exactement la même que celle de Boneh et Boyen dans [34]. Il s'agit de construire un algorithme $\mathcal{B}$ censé résoudre une instance aléatoire du problème $(q_\Sigma+1)$-GSDH : $(P, xP, x^2P,\ldots,x^{q_\Sigma+1}P)$. La seule différence réside dans la simulation des oracles de confirmation et de démenti. L'oracle décisionnel associé au problème $(q_\Sigma+1)$-GSDH permet, grâce aux points $xP,\ldots,x^{q_\Sigma+1}P$ de construire un oracle statique (voir [46]) $\mathcal{O}_x$ qui, étant donné Q et R en entrée répond si $R=xQ$. Cela permet de simuler parfaitement les protocoles.

□

Remarque 7 *On peut se passer de l'oracle décisionnel dans le théorème précédent. On se ramène alors au problème q-SDH classique, mais on perd un facteur* $2(q_C+q_D+1)$ *dans la réduction.*

2.6 Conclusion

Dans cette partie, nous avons contribué à l'étude des signatures indéniables en proposant une nouvelle propriété appelée conversion temporelle, et construit un nouveau schéma efficace de signatures indéniables totalement convertibles qui s'étend naturellement à des signatures convertibles dans le temps. Une amélioration possible serait de donner la possibilité au signataire de convertir toutes les signatures associées à une période de temps donnée, ce que ne permet pas notre schéma.

Par ailleurs, nous avons proposé un schéma de signatures indéniables dont la sécurité est prouvée dans le modèle standard. Ce schéma ne possède pas de propriétés de conversion totale, et les apporter à un schéma basé sur les couplages en conservant une sécurité dans le modèle standard est un problème ouvert.

Récemment, Ogata, Kurosawa et Heng [184] ont proposé un nouveau modèle d'attaquant pour les signatures indéniables : il est appelé “forge-et-usurpe”. Il ne va pas nécessairement essayer de forger, mais il va tenter de produire un semblant de signature, et de s'engager dans un protocole de confirmation avec un vérificateur. Il réussit son attaque s'il arrive à convaincre le vérificateur de la validité de sa “forge”. Pour éviter ce genre d'attaque, on pourrait imaginer que le signataire s'authentifie pendant les protocoles de confirmation et démenti, grâce à des techniques d'authentification déniable [83]

3

Signatures dirigées

Résumé : Nous proposons dans ce chapitre une étude formelle des signatures dirigées qui sont une extension des signatures indéniables, dans laquelle le signataire peut déléguer sa capacité à prouver qu'une signature est valide ou non à une tierce personne. Elles ont été introduites par 1993 par Lim et Lee [157]. Nous proposons le premier schéma de signatures dirigées totalement convertibles, et analysons sa sécurité dans le modèle de l'oracle aléatoire. Ce travail a fait l'objet d'une soumission en collaboration avec Pascal Paillier et Damien Vergnaud [142].

Sommaire

3.1 Introduction 111
3.1.1 Définition 112
3.1.2 Modèle de sécurité 114
3.1.3 Une nouvelle hypothèse calculatoire 117
3.2 Nouveau schéma 118
3.2.1 Description 118
3.2.2 Preuve de sécurité 119
3.2.3 Corollaire : une variante probabiliste de Boneh-Boyen 128
3.3 Conclusion 129

3.1 Introduction

Les signatures dirigées sont une extension des signatures indéniables. Elles font partie de la famille des schémas de signatures *déléguées*, dans lesquels une entité, autre que le signataire, mais éventuellement choisie par lui, a le pouvoir de vérifier par elle-même une signature provenant du signataire, sans interagir avec lui. Dans la suite, on appelera *confirmeur* cette entité. Par ailleurs, ses capacités peuvent être plus larges : il peut éventuellement être capable de transmettre sa conviction concernant la validité d'une signature à travers la participation à des protocoles de confirmation ou de démenti, il peut avoir le moyen de convertir une signature délégué en une signature classique ou de générer un reçu de conversion totale. Bien évidemment, il ne doit pas être capable de signer à la place du signataire. Suivant les applications pour lesquelles ces signatures déléguées sont destinées, il peut y avoir une disymétrie entre les pouvoirs du signataire et ceux du confirmeur. L'intérêt de ces signatures, outre le fait qu'elles soient des signatures indéniables, est qu'elles garantissent à quelqu'un en possession d'une signature, de pouvoir la vérifier si le signataire ne peut pas, ou refuse de participer aux protocoles de confirmation/démenti, en s'adressant au confirmeur.

Les *signatures dirigées* ont été introduites en 1993 par Lim et Lee [157, 158] et prévoient un rôle symétrique entre le signataire et le confirmeur. D'autres exemples classiques de ces signatures déléguées sont :

- les *signatures nominatives* introduites par Kim, Park et Won en 1994 [137, 138], et pour lesquelles, le "nominé" (qui n'est pas le signataire) est précisément le seul à pouvoir vérifier directement une signature, et aussi le seul à pouvoir participer à un protocole de confirmation/démenti pour prouver la validité/invalidité d'une signature ;
- les *signatures à confirmeur désigné* introduites par Chaum en 1994 [55], à propos desquelles Okamoto a montré en 1994 le résultat théorique suivant [186] : un schéma sûr de signature à confirmeur désigné existe si et seulement si il existe un algorithme sûr de chiffrement à clé publique. Les schémas pratiques proposés dans l'article ont été cryptanalysés par Michels et Stadler en 1998 [170]. Goldwasser et Waisbard ont proposé une transformation générique d'un schéma de signatures classiques en un schéma de signatures à confirmeur désigné en utilisant des preuves à témoin caché [87], et du chiffrement vérifiable [48] ;
- les *signatures à vérificateur limité* introduites par Araki, Uehara et Imamura [4] en 1999. Ce protocole a été ensuite cryptanalysé par Zhang et Kim [246] en 2003. Une version basée sur les couplages a été récemment proposée par Chen, Zhang et Kim dans [59] en 2004.

Parmi les schémas de signatures dirigées existants, aucun ne propose la génération de reçu de conversion totale. Dans le cas des signatures à vérificateur limité, le schéma de Araki *et al.* [4] l'offrait, mais il a ensuite été cassé.

Dans leur formalisation des signatures à confirmeur désigné, Goldwasser et Waisbard dans [118] ainsi que Camenisch et Michels dans [49] ne laissent la possibilité d'engager les protocoles de confirmation et démenti qu'au confirmeur, pour préserver le signataire. En effet, une personne malveillante pourrait vouloir forcer le signataire à prouver la validité ou l'invalidité d'une signature. Bien qu'un signataire puisse toujours prouver qu'il a bien généré une signature (par exemple en montrant une trace de la génération de la signature), il peut toujours, s'il ne peut *pas* prouver qu'une signature est valide, affirmer systématiquement qu'elle est invalide. Cette restriction vaut aussi pour les signatures nominatives. Encore une fois, ceci dépend essentiellement des applications dont elles vont faire l'objet. Les signatures dirigées offrent des possibilités similaires au signataire et au confirmeur. Les signatures déléguées sont utilisées, par exemple, dans des échanges de secret vérifiables [90], dans des signatures de contrat [5] ou dans des allocations de numéro d'enregistrement [149].

Dans la section suivante, nous proposons la première définition formelle des signatures dirigées, avec son modèle de sécurité (qui ne se trouvait pas dans le papier originel [157]), puis nous proposons le premier schéma de signatures dirigées ayant la propriété de conversion totale des signatures. Nous décrirons les preuves de sa sécurité dans le modèle de l'oracle aléatoire, sous de nouvelles hypothèses calculatoires présentées dans la section 3.1.3. Ces résultats ont fait l'objet d'une collaboration avec Pascal Paillier et Damien Vergnaud [142].

3.1.1 Définition

Idée générale L'idée des signatures dirigées universellement convertibles est donc la suivante : deux utilisateurs au sein d'une infrastructure à clé publique ont un rôle privilégié, à savoir un signataire A et un confirmeur B. A crée des signatures que seul B (et le signataire lui-même) peut vérifier. Bien sûr B ne peut produire des signatures à la place de A. Ces signatures sont

dites (A, B)-dirigées. Le signataire A et le confirmeur B peuvent alors tous deux
- prouver, de façon non-transférable, la validité ou l'invalidité de signatures (A, B)-dirigées à une tierce partie,
- convertir des signatures (A, B)-dirigées en des signatures classiques universellement vérifiables, sans qu'aucune autre signature (A, B)-dirigée n'en soit affectée,
- publier un reçu de conversion totale $\mathcal{R}$ qui transforme *toutes* les signatures (A, B)-dirigées en signatures universellement vérifiables, sans que cela n'ait d'impact sur toute autre signature (A', B')-dirigée, avec $(A', B') \neq (A, B)$.

Définition formelle Nous donnons maintenant une définition formelle des signatures dirigées.

Définition 48 (Signatures dirigées) *Soit k un entier. Un* schéma de signatures dirigées totalement convertibles *CDS de paramètre de sécurité k est composé des différents algorithmes et protocoles suivants :*

- génération des paramètres publics *CDS.Setup : algorithme probabiliste prenant en entrée le paramètre de sécurité k et renvoyant* les paramètres publics *;*
- génération de clés pour le signataire *CDS.SKeyGen : algorithme probabiliste prenant en entrée un entier k, des paramètres publics et renvoyant* une paire de clés publique et secrète *(pk_A, sk_A) ;*
- génération de clés pour le confirmeur *CDS.CKeyGen : algorithme probabiliste prenant en entrée un entier k, des paramètres publics et renvoyant une* paire de clés publique et secrète *(pk_B, sk_B) ;*

- signature *CDS.Sign : algorithme déterministe ou probabiliste prenant en entrée un entier k, des paramètres publics, une clé secrète sk, une clé publique pk_B et un message m, et renvoyant σ une* signature (A, B)-dirigées totalement convertible de m *(qu'on appellera* signature (A, B)-dirigées *dans la suite) ;*
- vérification par le signataire *CDS.SVerify : algorithme déterministe prenant en entrée un entier k, des paramètres publics, une clé secrète sk_A, une clé publique pk_B, un message m et une suite de bit σ et testant si σ est une signature (A, B)-dirigée valide du message m ;*
- vérification par le confirmeur *CDS.CVerify : algorithme déterministe prenant en entrée un entier k, des paramètres publics, une clé secrète sk_B, une clé publique pk_A, un message m et une suite de bit σ et testant si σ est une signature (A, B)-dirigée valide du message m ;*

- protocole de confirmation/démenti avec le signataire *CDS.$\{$SConfirm, SDeny$\}$: protocoles entre un signataire et une tierce partie, prenant en entrée un entier k, des paramètres publics, une clé secrète sk_A, une clé publique pk_B, un message m et une suite de bits σ. Sa sortie est une* preuve (éventuellement non-interactive) non transférable *que σ est une signature (A, B)-dirigée valide ou invalide de m ;*
- protocole de confirmation/démenti avec le confirmeur *CDS.$\{$CConfirm, CDeny$\}$: protocoles entre un confirmeur et une tierce partie, prenant en entrée un entier k, des paramètres publics, une clé secrète sk_B, une clé publique pk_A, un message m et une suite de bits σ. Sa sortie est une* preuve (éventuellement non-interactive) non transférable *que σ est une signature (A, B)-dirigée valide ou invalide de m ;*

- conversion individuelle par le signataire *CDS.ISConvert : algorithme déterministe prenant en entrée un entier k, des paramètres publics, une clé secrète sk_A, une clé publique pk_B, un*

message m et une suite de bits σ et testant si σ est une signature (A,B)-dirigée valide du message m. Si σ est valide, sa sortie est une signature (A,B)-dirigée A-convertie $\tilde{\sigma_A}$. *Si ce n'est pas le cas, il sort* `Invalide`*;*
- vérification d'une signature A-convertie *CDS.ISVerify : algorithme déterministe prenant en entrée un entier k, des paramètres publics, une clé publique pk_A, une clé publique pk_B, un message m et une suite de bits $\tilde{\sigma_A}$ et testant si $\tilde{\sigma_A}$ est une signature (A,B)-dirigée A-convertie valide du message m.*
- conversion individuelle par le confirmeur *CDS.ICConvert : algorithme déterministe prenant en entrée un entier k, des paramètres publics, une clé secrète sk_B, une clé publique pk_A, un message m et une suite de bits σ et testant si σ est une signature (A,B)-dirigée valide du message m. Si σ est valide, sa sortie est une* signature (A,B)-dirigée B-convertie $\tilde{\sigma_B}$. *Si ce n'est pas le cas, il sort* `Invalide`*;*
- vérification d'une signature B-convertie *CDS.ICVerify : algorithme déterministe prenant en entrée un entier k, des paramètres publics, une clé publique pk_A, une clé publique pk_B, un message m et une suite de bits $\tilde{\sigma_B}$ et testant si $\tilde{\sigma_B}$ est une signature (A,B)-dirigée B-convertie valide du message m.*

- génération de reçu de conversion totale par le signataire *CDS.TSReceipt : algorithme déterministe prenant en entrée un entier k, des paramètres publics, une clé secrète sk_A et une clé publique pk_B, et renvoyant un* reçu de conversion totale $\mathcal{I}_A$ *;*
- génération de reçu de conversion totale par le confirmeur *CDS.TCReceipt : algorithme déterministe prenant en entrée un entier k, des paramètres publics, une clé secrète sk_B et une clé publique pk_A, et renvoyant un* reçu de conversion totale $\mathcal{I}_B$ *;*
- vérification d'une signature avec reçu de conversion totale *CDS.{TSVerify, TCVerify} : algorithmes déterministes prenant en entrée un entier k, des paramètres publics, une clé publique pk_A, une clé publique pk_B, un message $m \in \{0,1\}^*$ et deux suites de bits σ et $\mathcal{I}$, et testant si $\mathcal{I}$ est un reçu de conversion totale valide du signataire ou du vérificateur. Si ce n'est pas le cas, il indique* `Reçu invalide` *et sinon il vérifie que σ est une signature (A,B)-dirigée valide de m. Si elle ne l'est pas, il renvoie* `Invalide`*;*

3.1.2 Modèle de sécurité

Dans cette section, nous donnons les définitions formelles des propriétés de sécurité requises pour les signatures dirigées. On note

$$\mathsf{CDS} = (\mathsf{Setup}, \mathsf{SKeyGen}, \mathsf{VKeyGen}, \mathsf{Sign}, \mathsf{SVerify}, \mathsf{CVerify}, \mathsf{SConfirm}, \mathsf{SDeny}, \mathsf{CConfirm}, \mathsf{CDeny},$$

$$\mathsf{ISConvert}, \mathsf{ISVerify}, \mathsf{ICConvert}, \mathsf{ICVerifyTSReceipt}, \mathsf{TCReceipt}, \mathsf{TSVerify}, \mathsf{TCVerify})\,.$$

3.1.2.1 Consistance et significativité

La consistance des signatures dirigées totalement convertibles est définie de la même façon que pour les signatures indéniables : une signature correctement formée sera acceptée par l'algorithme de vérification. De même, les protocoles de confirmation et de démenti doivent être consistants et significatifs.

3.1.2.2 Inforgeabilité

L'inforgeabilité des signatures dirigées est naturellement reliée à celle des signatures indéniables. Dans le cas des signatures dirigées, l'attaquant peut être éventuellement n'importe lequel des confirmeurs choisis par le signataire. Il a alors plus de connaissance qu'un utilisateur lambda. Aussi, on doit considérer que l'attaquant connaît la clé secrète des confirmeurs du signataire. Il peut donc interroger un oracle de signatures dirigées vers un confirmeur de son choix, dont il connait la clé secrète. En outre, l'attaquant a accès à un oracle de conversion individuelle qu'il peut interroger sur toute signature de son choix. On suppose aussi qu'il connaît les reçus de conversion totale du signataire pour n'importe quel confirmeur. Comme toujours, la signature retournée par l'attaquant ne doit pas avoir été obtenue auprès de l'oracle de signature. Il peut aussi s'engager dans des protocoles de confirmation ou démenti avec le signataire. Pour récapituler, l'attaquant peut interroger

- un oracle de signature Σ auquel il demande une signature (A, B)-dirigée d'un message quelconque, où A est le signataire attaqué, et B un confirmeur de son choix,
- un oracle de conversion individuelle Υ d'une signature (A, B)-dirigée quelconque, pour un confirmeur B de son choix
- un oracle de génération de reçu de conversion totale $\mathfrak{U}$ (lié lui aussi à un confirmeur qu'il a choisi)
- les oracles $\mathfrak{C}$ et $\mathfrak{D}$ de confirmation et de démenti
- un oracle d'enregistrement de clés publiques $\mathfrak{R}$, duquel il obtient un certificat associé à une clé publique qu'il a choisie, dont il connaît le secret associé, et qui correspond à une clé d'un confirmeur potentiel.

Sa forge est un couple (m, σ) accompagné d'une clé publique pk_B d'un confirmeur B, σ est une signature (A, B)-dirigée.

La définition formelle est alors :

Définition 49 (Inforgeabilité) *Soit k un entier et CDS un schéma de signatures dirigées totalement convertibles de paramètre de sécurité k. Soit $\mathcal{A}$ un adversaire $\mathsf{EF\text{-}CMA}$ contre CDS. Considérons l'expérience aléatoire suivante :*

$$\boxed{\textit{Experiment } \mathbf{Exp}^{\mathsf{ef-cma}}_{\mathsf{CDS},\mathcal{A}}(k)}$$

$$\begin{array}{l}
\text{params} \leftarrow \mathsf{CDS.Setup}(k) \\
(pk_A, sk_A) \leftarrow \mathsf{CDS.SKeyGen}(k, \text{params}) \\
(m, \sigma, pk_B) \leftarrow \mathcal{A}^{\Sigma,\Upsilon,\mathfrak{U},\mathfrak{C},\mathfrak{D},\mathfrak{R}}(k, \text{params}, pk_A) \\
\textit{Retourne } \mathsf{CDS.SVerify}(k, \text{params}, sk_A, pk_B, m, \sigma,)
\end{array}$$

On définit le succès de l'adversaire $\mathcal{A}$ comme étant

$$\mathbf{Succ}^{\mathsf{ef\text{-}cma}}_{\mathsf{CDS},\mathcal{A}}(k) = \Pr\left[\mathbf{Exp}^{\mathsf{ef\text{-}cma}}_{\mathsf{CDS},\mathcal{A}}(k) = 1\right].$$

Soit τ un entier et ε un réel dans $[0, 1]$. Le schéma CDS est dit (k, τ, ε)-$\mathsf{EF\text{-}CMA}$ si aucun adversaire $\mathsf{EF\text{-}CMA}$ $\mathcal{A}$ fonctionnant en temps τ n'a un succès $\mathbf{Succ}^{\mathsf{ef\text{-}cma}}_{\mathsf{CDS},\mathcal{A}}(k) \geq \varepsilon$.

3.1.2.3 Invisibilité

Dans cette partie, nous introduisons formellement la notion d'*invisibilité* pour les signatures dirigées. Nous aurions pu considérer la notion d'anonymat, mais comme l'ont montré Galbraith et Mao dans [95], les notions sont essentiellement équivalentes. Par ailleurs, nous déclinons l'invisibilité sous deux formes différentes, selon les capacités de l'attaquant. Celui-ci fonctionne en deux étapes : dans sa premiére étape find, il reçoit les clés publiques d'un signataire cible pk_A et d'un confirmeur cible pk_B et sort un message $m^\star$, ainsi qu'une information d'état $\mathcal{I}^\star$. Dans l'étape suivante, l'attaquant prend en entrée $\mathcal{I}^\star$ et une signature challenge $\sigma^\star$ obtenue auprès de l'oracle de signature sur $m^\star$, ou en prenant une signature au hasard dans l'espace des signatures (A, B)-dirigées. L'attaquant devine alors si $\sigma^\star$ est une signature (A, B)-dirigée valide de $m^\star$.

Dans le scénario de l'attaque pour lever l'invisibilité faible (modèle weak-Inv-CMA), l'attaquant a accès, dans ces deux étapes, à l'oracle de signature Σ, et aux oracles de confirmation et démenti $\mathfrak{C}$ et $\mathfrak{D}$. Dans le modèle strong-Inv-CMA, on permet à l'attaquant d'accéder aussi à un oracle de conversion individuelle Υ, et à un oracle de génération de reçu de conversion totale $\mathfrak{U}$. Dans tous les cas, l'attaquant peut interroger ces oracles sur tous les messages et confirmeur de son choix, avec pour seule restriction de ne pas demander de conversion individuelle ou de confirmation/démenti sur $(m^\star, \sigma^\star)$, et de ne pas demander de reçu de conversion totale à Υ pour des signatures (A, B)-dirigées.

Invisibilité faible L'étude de l'invisibilité faible se justifie par une utilisation possible des signatures dirigées sans la propriété de conversion. La définition formelle est la suivante :

Définition 50 (Invisibilité faible) *Soit k un entier et CDS un schéma de signatures dirigées totalement convertibles de paramètre de sécurité k. Soit $\mathcal{A}$ un adversaire weak-Inv-CMA contre CDS. Considérons l'expérience aléatoire suivante, pour un $r \in \{0, 1\}$:*

$\boxed{\textit{Experiment } \mathbf{Exp}^{\textsf{weak-inv-cma-r}}_{\textsf{CDS},\mathcal{A}}(k)}$
params $\leftarrow$ $\textsf{CDS.Setup}(k)$
$(pk_A, sk_A) \leftarrow \textsf{CDS.SKeyGen}(k, \text{params})$
$(pk_B, sk_B) \leftarrow \textsf{CDS.CKeyGen}(k, \text{params})$
$(m^\star, \mathcal{I}^\star) \leftarrow \mathcal{A}^{\Sigma,\mathfrak{C},\mathfrak{D},\mathfrak{R}}(\textsf{find}, k, \text{params}, pk_A, pk_B)$
Si $\mathsf{r} = 0$ *alors* $\sigma^\star \leftarrow \textsf{CDS.Sign}(k, \text{params}, sk_A, pk_B, m^\star)$
Si $\mathsf{r} = 1$ *alors* $\sigma^\star \xleftarrow{R} \mathcal{S}ign$
$d \leftarrow \mathcal{A}^{\Sigma,\mathfrak{C},\mathfrak{D},\mathfrak{R}}(\textsf{guess}, \mathcal{I}^\star, k, \text{params}, m^\star, \sigma^\star, pk_A, pk_B)$
Retourne d

*On définit l'*avantage *de l'attaquant $\mathcal{A}$ comme étant*

$$\mathbf{Adv}^{\textsf{weak-inv-cma}}_{\textsf{CDS},\mathcal{A}}(k) = \left|\Pr\left[\mathbf{Exp}^{\textsf{weak-inv-cma-1}}_{\textsf{CDS},\mathcal{A}}(k) = 1\right] - \Pr\left[\mathbf{Exp}^{\textsf{weak-inv-cma-0}}_{\textsf{CDS},\mathcal{A}}(k) = 1\right]\right|.$$

Soit τ un entier et ε un réel dans $[0, 1]$. Le schéma CDS est dit (k, τ, ε)-weak-Inv-CMA (ou faiblement invisible) si aucun adversaire weak-Inv-CMA $\mathcal{A}$, fonctionnant en temps τ, n'a un avantage $\mathbf{Adv}^{\textsf{weak-inv-cma}}_{\textsf{CDS},\mathcal{A}}(k) \geq \varepsilon$.

Nous montrerons que les signatures décrites dans la figure 3.2.1 sont invisibles en ce sens sous le problème xyz-DDH de la définition 37.

Invisibilité forte L'invisibilité forte est une exigence forte de sécurité pour les signatures dirigées. L'attaquant a accès à tous les oracles pendant son attaque, contrairement au scenario de l'attaque contre l'invisibilité faible. La définition formelle est la suivante.

Définition 51 (Invisibilité forte) *Soit k un entier et CDS un schéma de signatures dirigées de paramètre de sécurité k. Soit $\mathcal{A}$ un adversaire strong-Inv-CMA contre CDS. Considérons l'expérience aléatoire suivante, pour un $r \in \{0,1\}$:*

$$\boxed{\textit{Experiment } \mathbf{Exp}^{\textsf{strong-inv-cma-}r}_{\mathit{CDS},\mathcal{A}}(k)}$$
$$\begin{array}{l}
\text{params} \leftarrow \mathit{CDS.Setup}(k) \\
(pk_A, sk_A) \leftarrow \mathit{CDS.SKeyGen}(k, \text{params}) \\
(pk_B, sk_B) \leftarrow \mathit{CDS.CKeyGen}(k, \text{params}) \\
(m^\star, \mathcal{I}^\star) \leftarrow \mathcal{A}^{\Sigma,\mathfrak{C},\mathfrak{D},\Upsilon,\mathfrak{U},\mathfrak{R}}(\mathit{find}, k, \text{params}, pk_A, pk_B) \\
\textit{Si } \mathsf{r} = 0 \textit{ alors } \sigma^\star \leftarrow \mathit{CDS.Sign}(k, \text{params}, sk_A, pk_B, m^\star) \\
\textit{Si } \mathsf{r} = 1 \textit{ alors } \sigma^\star \xleftarrow{R} \mathcal{S}ign \\
d \leftarrow \mathcal{A}^{\Sigma,\mathfrak{C},\mathfrak{D},\Upsilon,\mathfrak{U},\mathfrak{R}}(\mathit{guess}, \mathcal{I}^\star, k, \text{params}, m^\star, \sigma^\star, pk_A, pk_B) \\
\textit{Retourner } d
\end{array}$$

*On définit l'*avantage *de l'attaquant $\mathcal{A}$ comme étant*

$$\mathbf{Adv}^{\textsf{strong-inv-cma}}_{\mathit{CDS},\mathcal{A}}(k) = \left| \Pr\left[\mathbf{Exp}^{\textsf{strong-inv-cma-1}}_{\mathit{CDS},\mathcal{A}}(k) = 1\right] - \Pr\left[\mathbf{Exp}^{\textsf{strong-inv-cma-0}}_{\mathit{CDS},\mathcal{A}}(k) = 1\right] \right|.$$

Soit τ un entier et ε un réel dans $[0,1]$. Le schéma CDS est dit (k,τ,ε)-strong-Inv-CMA (ou fortement invisible) si aucun adversaire strong-Inv-CMA $\mathcal{A}$, fonctionnant en temps τ, n'a un avantage $\mathbf{Adv}^{\textsf{strong-inv-cma}}_{\mathit{CDS},\mathcal{A}}(k) \geq \varepsilon$.

Nous montrerons que l'invisibilité forte de notre schéma DSBM repose sur un problème non-standard, et dépendant du nombre de requêtes à l'oracle de signature : q_Σ-xyz-DCAA de la définition 39.

3.1.3 Une nouvelle hypothèse calculatoire

Diffie-Hellman carré flexible Nous introduisons ici un problème calculatoire nouveau qui semble difficile (on peut donner le même genre de bornes sur sa complexité dans le modèle générique comme pour xyz-DDH), et il paraît plus naturel qu'un certain nombre des problèmes déjà exposés.

Problème Diffie-Hellman carré flexible [FSDH] : soit $\mathbb{G}$ un groupe d'ordre un premier q engendré par P. Étant donné un couple (P, xP), calculer un triplet (Q, xQ, x^2Q), pour un Q librement choisi dans $\mathbb{G}$.

La définition suivante quantifie l'hypothèse associée :

Définition 52 (FSDH) *Soit BDHGen un générateur de groupes BDH symétrique et $\mathcal{D}$ un attaquant qui prend en entrée un paramètre de sécurité, un quintuplet généré par BDHGen et un couple $(P, xP) \in \mathbb{G}^2$, où $x \in [\![1, q-1]\!]$ et renvoie un triplet $(Q, Q', Q'') \in \mathbb{G}^3$. Considérons les expériences aléatoires suivantes, où k est un paramètre de sécurité :*

$$\boxed{\textit{Experiment } \mathbf{Exp}^{\textsf{fsdh}}_{\mathit{BDHGen},\mathcal{D}}(k)}$$
$$\begin{array}{l}
(q, P, \mathbb{G}, \mathbb{H}, e) \leftarrow \mathit{BDHGen}(k) \\
\text{setup} \leftarrow (q, P, \mathbb{G}, \mathbb{H}, e) \\
x \xleftarrow{R} [\![1, q-1]\!] \\
(Q, Q', Q'') \leftarrow \mathcal{D}(k, \text{setup}, (P, xP)) \\
\textit{Retourner } 1 \textit{ if } Q' = xQ \textit{ et } Q'' = x^2Q, \ 0 \textit{ sinon}
\end{array}$$

on définit alors le succès de $\mathcal{D}$ pour résoudre le problème Diffie-Hellman carré flexible par

$$\mathbf{Succ}^{\mathsf{fsdh}}_{\mathsf{BDHGen},\mathcal{D}}(k) = \Pr\left[\mathbf{Exp}^{\mathsf{fsdh}}_{\mathsf{BDHGen},\mathcal{D}}(k) = 1\right]$$

Étant donné $(k, \tau) \in \mathbb{N}^2$ et $\varepsilon \in [0, 1]$, le problème FSDH est dit (k, τ, ε)-difficile s'il n'existe pas d'attaquant $\mathcal{D}$ qui, en temps τ, a un succès $\mathbf{Succ}^{\mathsf{fsdh}}_{\mathsf{BDHGen},\mathcal{D}}(k) \geq \varepsilon$.

Remarque 8 *L'inforgeabilité du schéma DSBM proposé dans la section suivante repose sur le problème FSDH. Contrairement à l'invisibilité forte de DSBM, ou plus généralement à la sécurité des schémas reposant sur les familles des problèmes ℓ-CAA ou ℓ-SDH, l'inforgeabilité de DSBM repose sur une instance qui ne dépend pas du nombre de requêtes à l'oracle de signature q_Σ. Par ailleurs, sous une hypothèse appelée KEA1, le problème FSDH est équivalent au problème CDH classique. L'hypothèse KEA1 a été introduite en 1991 par Damgård [69] pour étudier la sécurité de schémas de chiffrement. KEA et des variantes, préalablement présentées erronées dans [120], ont été formellement étudiées récemment par Bellare et Palacio dans [22], dans le cadre d'une étude de protocoles zero-knowledge en trois passes. Intuitivement, cette hypothèse consiste à supposer que si un algorithme est capable, étant donné une paire (P, xP) de calculer une autre paire (Q, xQ), alors il "doit connaître" $\log_P(Q)$, et donc il a calculé en fait $(\log_P(Q)P, \log_P(Q)(xP))$ pour obtenir la nouvelle paire.*

3.2 Nouveau schéma

Le schéma que nous proposons dans cette partie est le premier schéma de signatures dirigées qui offre la propriété de conversion totale. Il est basé sur l'utilisation d'un application bilinéaire admissible, et les protocoles de confirmation et de démenti sont des preuves classiques d'appartenance à un langage. C'est une extention naturelle du schéma de signatures indéniables CUSBM présenté précédemment. L'idée est de remplacer dans ce schéma la valeur $a_2H(m||r)$ par rP_B où P_B est la clé publique du confirmeur.

3.2.1 Description

Nous décrivons dans cette partie le schéma DSBM (pour Directed Signatures based on Bilinear Map). La figure 3.1 décrit les algorithmes de génération des paramètres publics, de génération des paires de clés, et de signature. Les algorithmes de vérification de la signature par le signataire et le confirmeur sont décrits dans la figure 3.2, les protocoles de confirmation et de démenti dans la figure 3.3, et les algorithmes de conversion individuelle dans la figure 3.4. Finalement, les générations de reçus de conversion totale sont présentées dans la figure 3.5.

Une signature (A, B)-dirigée est de la forme

$$\sigma = \left(rP, \frac{rx_1}{x_2 + h(m||rP||P_B)}P_B\right),$$

soit deux points sur une courbe, ce qui donne une longueur de signature d'essentiellement $2\log_2(q)$ bits. La génération de la signature ne nécessite pas de calcul de couplage, seulement quelques opérations dans le corps de base et 2 multiplications scalaires. Les vérifications de la signature nécessitent quant à elles des calculs de couplages. Les opérations de conversion ne sont par contre que de simples multiplications scalaires. On peut par ailleurs construire une variante ayant un compromis calcul en ligne/hors ligne particulièrement intéressant grâce à une astuce remarquée par Libert et Quisquater dans [156] pour un schéma de *signcryption* et pour les signatures de Boneh-Boyen originelles. En rallongeant la clé publique par un troisième point : $pk = (P_1, P_2, P_3)$ et $sk = (x_1, x_2, x_3)$ les logarithmes discrets des P_i en base P, on pose

$$\sigma = \left(rP, \frac{rx_1}{x_2 + h(m||rP||P_B) + x_3 s} P_B, s \right)$$

avec r et s des aléas dans $[\![1, q-1]\!]$. La vérification de la signature $\sigma = (Q_1, Q_2, s)$ est alors

$$e(Q_2, P_{A,2} + h(m||Q_1||P_B)P + sP_{A,3}) = e(Q_1, P_{A,1})^y \quad \text{ou} \quad e(Q_1, P_B)^{x_1}$$

selon le vérificateur (le signataire ou le confirmeur). Les autres procédures du schéma se déduisent aisément en intégrant la troisième clé et l'aléa s.

Grâce à cette construction, le signataire calcule hors ligne $(Q_1, Q_2) = (rP, rx_1(x_2 + z)P_B)$ où z est un élément pris au hasard dans $[\![1, q-1]\!]$. Étant donné un message m à signer, le signataire complète la signature en calculant en ligne

$$s = x_3^{-1}(z - h(m||Q_1||P_B))$$

(x_3^{-1} peut être précalculée). Cette modification n'altère pas la sécurité du schéma.

***Remarque* 9 (Conversion dans le temps)** *Comme pour les signatures indéniables, il est possible d'adapter ce schéma pour que les signatures deviennent convertibles dans le temps.*

3.2.2 Preuve de sécurité

Dans cette section, nous étudions la sécurité de DSBM dans le modèle de l'oracle aléatoire. En premier lieu, nous montrons que l'inforgeabilité du schéma est reliée à FSDH, l'invisibilité faible au problème xyz-DDH, et l'invisibilité forte au problème ℓ-xyz-DCAA.

Il apparaît dans les preuves que les deux clés secrètes ont des rôles bien distincts : a_2 peut être vu comme une clé de signature, et a_1 comme une clé d'anonymat.

3.2.2.1 Inforgeabilité

L'inforgeabilité du schéma DSBM proposé est reliée à la difficulté du problème Diffie-Hellman carré flexible. C'est ce que nous montrons dans le théorème suivant.

Théorème 17 (Inforgeabilité de DSBM) *Soit BDHGen un générateur de groupes BDH symétrique et soit DSBM le schéma de signatures dirigées associé. Dans le modèle de l'oracle aléatoire, on suppose qu'il existe un adversaire EF-CMA $\mathcal{A}$ contre DSBM, de paramètre de sécurité k, ayant un succès $\mathbf{Succ}^{\mathsf{ef-cma}}_{\mathsf{DSBM},\mathcal{A}}(k) = \varepsilon$ en temps τ, et pouvant faire $q_{\mathcal{H}}$ requêtes à l'oracle aléatoire, q_Σ requêtes à l'oracle de signature et q_Υ requêtes à l'oracle de conversion individuelle, $q_{\mathfrak{C}}$ et $q_{\mathfrak{D}}$ requêtes aux oracles de conversion et de démenti, $q_{\mathfrak{U}}$ requêtes à l'oracle de génération de reçu de conversion totale et $q_{\mathfrak{R}}$ requêtes à un oracle d'enregistrement de clé. On suppose par ailleurs que $\varepsilon \geq q_{Tot}2^{1-k}$, où $q_{Tot} = q_{\mathcal{H}} + q_\Sigma + q_\Upsilon + q_{\mathfrak{C}} + q_{\mathfrak{D}} + 1$. Alors il existe un adversaire $\mathcal{D}$ contre FSDH, de succès $\mathbf{Succ}^{\mathsf{fsdh}}_{\mathsf{BDHGen},\mathcal{D}}(k) = \varepsilon'$ en temps τ' tel que*

$$\begin{cases} \varepsilon' \geq \dfrac{1}{q_{Tot}} \left(\dfrac{\varepsilon}{2} - \dfrac{q_{Tot}}{2^k} \right)^2 \\ \tau' \leq 2q_{\mathfrak{R}}\tau + O(1). \end{cases}$$

Génération des paramètres publics : $\mathsf{DSBM.Setup}$

• k est un paramètre de sécurité,
• $(q, P, \mathbb{G}, \mathbb{H}, e)$ est généré par $\mathsf{BDHGen}(k)$
• Soit $[\{0,1\}^* \times \mathbb{G}^2 \longrightarrow [\![1, q-1]\!]]$ une famille de fonctions de hachage h un élément pris au hasard dans cette famille.

Génération des clés pour le signataire : $\mathsf{DSBM.SKeyGen}$

• $(a_1, a_2) \in [\![1, q-1]\!]^2$ est la clé secrète, (P_1, P_2) est la clé publique, avec $P_1 = a_1 P$ et $P_2 = a_2 P$

Génération des clés pour le confirmeur : $\mathsf{DSBM.CKeyGen}$

• $b \in [\![1, q-1]\!]$ est la clé secrète, $P_B = bP$ est la clé publique

Signature : $\mathsf{DSBM.Sign}$

• Étant donné un message m, les clés secrètes de signature a_1 et a_2, et la clé publique du confirmeur P_B, Alice tire au hasard un elément $r \in \{0,1\}^{n_r}$ et calcule $Q_1 = rP$ et $t = a_2 + h(m||Q_1||P_B)$. Si $t = 0 \mod q$, elle tire un nouveau r au hasard, sinon, elle pose

$$Q_2 = \frac{a_1 r}{a_2 + h(m||Q_1||P_B)} P_B$$

• la signature du message m est $\sigma = (Q_1, Q_2)$.

FIG. 3.1 – DSBM : description de la signature

PREUVE :

Soit k un paramètre de sécurité, et $(q, P, \mathbb{G}, \mathbb{H}, e)$ un 5-uplet généré par $\mathsf{BDHGen}(k)$. Considérons une instance aléatoire (P, xP) du problème FSDH. Nous allons construire un algorithme de réduction $\mathcal{D}$ qui, en interagissant avec un attaquant EF-CMA $\mathcal{A}$ contre DSBM, va produire un triplet (Q, xQ, x^2Q), où $Q \in \mathbb{G}$. Cette preuve est basée sur le *forking lemma* [199].

On considère donc $\mathcal{A}$ qui produit une forge avec un succès $\mathbf{Succ}^{\mathsf{ef-cma}}_{\mathsf{DSBM},\mathcal{A}}(k) = \varepsilon$ en temps τ. $\mathcal{D}$ interagit avec $\mathcal{A}$ de la façon suivante. Tout d'abord, il tire au hasard $x_1 \in [\![1, q-1]\!]$, et pose $P_1 = x_1 P$, et $P_2 = xP$.

SIMULATIONS DANS L'ÉTAPE Find. Tout d'abord, on définit une sous-routine $\mathcal{D}_0(\omega)$ de $\mathcal{D}$ qui, étant donné un élément aléatoire ω, lance $\mathcal{A}$ avec ω comme bande aléatoire, lui transmet $(q, P, \mathbb{G}, \mathbb{H}, e)$ comme paramètres publics, et $pk_A = (P_1, P_2)$ comme clé publique à attaquer. Maintenant, $\mathcal{D}_0(\omega)$ simule les différents oracles.

– **Simulation de $\mathcal{H}$:** pour tout message $m \in \{0,1\}^*$ proposé à l'oracle aléatoire avec le couple $(Q_1, P_B) \in \mathbb{G}^2$, $\mathcal{D}_0(\omega)$ renvoie la valeur $\mathcal{H}(m||Q_1||P_B)$ si celle-ci est déjà définie,

Vérification par le confirmeur et le signataire :

– DSBM.CVerify
  - Étant donné un message $m \in \{0,1\}^*$ et une signature $\sigma = (Q_1, Q_2)$, le confirmeur vérifie grâce à sa clé secrète b que

$$e(Q_2, P_2 + h(m||Q_1||P_B)P) = e(Q_1, P_1)^b$$

– DSBM.SVerify
  - Étant donné un message $m \in \{0,1\}^*$ et une signature $\sigma = (Q_1, Q_2)$, le signataire vérifie grâce à sa clé secrète a_1 que

$$e(Q_2, P_2 + h(m||Q_1||P_B)P) = e(Q_1, P_B)^{a_1}$$

Fig. 3.2 – DSBM : description de la vérification

Protocoles de confirmation et de démenti :

DSBM.{SConfirm,SDeny}
• Étant donné un message m et une signature $\sigma = (Q_1, Q_2)$, Alice confirme celle-ci grâce à la preuve (éventuellement non-interactive) de connaissance suivante :

$$\begin{array}{rl} & DVP(a_1 : e(Q_2, P_2 + h(m||Q_1||P_B)P) = e(Q_1, P_B)^{a_1} \wedge P_1 = a_1 P) \\ \text{ou} & DVP(a_1 : e(Q_2, P_2 + h(m||Q_1||P_B)P) \neq e(Q_1, P_B)^{a_1} \wedge P_1 = a_1 P) \end{array}$$

DSBM.{CConfirm,CDeny}
• Étant donné un message m et une signature $\sigma = (Q_1, Q_2)$, Bob confirme celle-ci grâce à la preuve (éventuellement non-interactive) de connaissance suivante :

$$\begin{array}{rl} & DVP(b : e(Q_2, P_2 + h(m||Q_1||P_B)P) = e(Q_1, P_1)^{b} \wedge P_B = bP) \\ \text{ou} & DVP(b : e(Q_2, P_2 + h(m||Q_1||P_B)P) \neq e(Q_1, P_1)^{b} \wedge P_1 = a_1 P) \end{array}$$

Fig. 3.3 – DSBM : description des protocoles de confirmation et démenti

sinon, il tire au hasard un élément $h \in [\![1, q-1]\!]$ qu'il renvoie comme haché en maintenant une H-List à jour.

– **Simulation de Σ :** étant donné un message $m \in \{0,1\}^*$ à signer en direction de B, et une clé publique P_B du confirmeur B, $\mathcal{D}_0(\omega)$ tire au hasard $(r, h) \in [\![1, q-1]\!]^2$, et pose $Q_2 = x_1 r P_B$ et $Q_1 = rP_2 + rhP$. Si $\mathcal{H}(m||Q_1||P_B)$ est déjà définie et est différente de h, $\mathcal{D}_0(\omega)$ est relancé avec une nouvelle valeur de r. Sinon, il pose $\mathcal{H}(m||Q_1||P_B) = h$, met à jour la H-List, et sort $\sigma = (Q_1, Q_2)$ comme signature, puisqu'on a bien alors

$$e(Q_2, xP + hP) = e(x_1 r P_B, xP + hP) = e(rP_2 + rhP, x_1 P_B) = e(Q_1, P_B)^{x_1}.$$

Conversions individuelles et vérifications :

- DSBM.ISConvert
  • Étant donné un message m et une signature potentielle (Q_1, Q_2), Alice vérifie la signature. Si elle est invalide, elle répond `Invalide`, sinon, elle calcule le point $Q_3 = a_1 Q_1$ et la signature A-convertie de m est $\tilde{\sigma_A} = (Q_1, Q_2, Q_3)$.
- DSBM.ISVerify
  • Étant donné un message m et une signature A-convertie (Q_1, Q_2, Q_3), un utilisateur quelconque accepte la signature si et seulement si

$$e(Q_1, P_1) = e(Q_3, P) \text{ et } e(Q_2, P_2 + h(m||Q_1||P_B)P) = e(Q_3, P_B).$$

- DSBM.ICConvert
  • Étant donné un message m et une signature potentielle (Q_1, Q_2), Bob vérifie la signature. Si elle est invalide, elle répond `Invalide`, sinon, il calcule le point $Q_3 = bQ_1$ et la signature B-convertie de m est $\tilde{\sigma_B} = (Q_1, Q_2, Q_3)$.
- DSBM.ICVerify
  • Étant donné un message m et une signature B-convertie (Q_1, Q_2, Q_3), un vérificateur quelconque accepte la signature si et seulement si

$$e(Q_1, P_B) = e(Q_3, P) \text{ et } e(Q_2, P_2 + h(m||Q_1||P_B)P) = e(Q_3, P_1).$$

Fig. 3.4 – DSBM : description des protocoles de conversion individuelle et de vérification

Générations des reçus de conversion totale et vérifications :

- DSBM.{TSReceipt,TCReceipt}
  Alice ou Bob publie le point

$$I = bP_1 = a_1 P_B = a_1 bP$$

- DSBM.{TSVerify,TCVerify}
  Étant donné un message m et une signature $\sigma = (Q_1, Q_2)$, un vérificateur quelconque accepte la signature si et seulement si

$$e(P_1, P_B) = e(I, P) \text{ et } e(Q_2, P_2 + h(m||Q_1||P_B)P) = e(Q_1, I).$$

Fig. 3.5 – DSBM : description de la génération du reçu de conversion totale et vérification

- **Simulation de $\mathfrak{C}$ et $\mathfrak{D}$:** Les protocoles de confirmation et de démenti peuvent être exécutés par $\mathcal{D}_0(\omega)$ puisqu'il connaît la clé x_1, et peut donc participer aux preuves

$$\text{DVPK}(x_1 : e(Q_2, P_2 + h(m||Q_1||P_B)P) = e(Q_1, P_B)^{x_1} \wedge P_1 = x_1 P)$$

et

$$\text{DVPK}(x_1 : e(Q_2, P_2 + h(m||Q_1||P_B)P) \neq e(Q_1, P_B)^{x_1} \wedge P_1 = x_1 P),$$

pour toute signature $\sigma = (Q_1, Q_2)$ de tout message $m \in \{0,1\}^*$ vers tout P_B. Au cours de la simulation de chacune de ces preuves, il faudra faire appel à une simulation de l'oracle aléatoire.

- **Simulation de Υ :** étant donné la clé publique P_B d'un confirmeur, un message $m \in \{0,1\}^*$ et $\sigma = (Q_1, Q_2)$, $\mathcal{D}_0(\omega)$ fait appel à la simulation de $\mathcal{H}$ pour vérifier, grâce à sa connaissance de x_1, que
$$e(Q_2, P_2 + \mathcal{H}(m||Q_1||P_B)P) = e(Q_1, P_B)^{x_1}.$$
Si ce n'est pas le cas, $\mathcal{D}_0(\omega)$ répond `Invalide`, sinon, il renvoie $\tilde{\sigma}_A = (Q_1, Q_2, x_1 Q_1)$.

- **Simulation de $\mathfrak{U}$:** étant donné la clé P_B d'un confirmeur, $\mathcal{D}_0(\omega)$ produit $I = x_1 P_B$ grâce à sa connaissance de x_1.

Si $\mathcal{A}$ sort une forge $((m, \sigma), P_B)$ avec $\sigma = (Q_1, Q_2)$, $\mathcal{D}_0(\omega)$ simule une nouvelle fois $\mathcal{H}$ pour obtenir $h = \mathcal{H}(m||Q_1||P_B)$ et vérifie que $e(Q_2, P_2 + \mathcal{H}(m||Q_1||P_B)P) = e(Q_1, P_B)^{x_1}$. $\mathcal{D}_0(\omega)$ a gagné si cette égalité est vraie et si (Q_1, Q_2) n'apparaît pas dans les traces de la simulation de l'oracle de signature Σ.

L'algorithme $\mathcal{D}$ lance $\mathcal{D}_0(\omega)$ avec des bandes aléatoires ω, jusqu'à ce qu'il gagne. On note $((m, \sigma), P_B)$ la forge victorieuse de $\mathcal{A}$, où $\sigma = (Q_1, Q_2)$. $\mathcal{D}$ mémorise alors ω, l'indice α de la requête (m, Q_1) dans la H-List, ainsi que les α premières reponses de $\mathcal{H}$, notées $h_1, \ldots, h_\alpha$. Si β est l'indice, dans la liste des signatures demandées à l'oracle Σ, de sa dernière sortie *avant* que $\mathcal{H}$ retourne h_α, alors $\mathcal{D}$ retient aussi $\beta, \sigma_1, \ldots, \sigma_\beta$.

REJEU. Pour pouvoir utiliser le *forking lemma*, il est nécessaire de construire une deuxième sous-routine probabiliste $\mathcal{D}_1(\omega)$ qui rejoue la dernière exécution victorieuse de $\mathcal{D}_0(\omega)$ jusqu'au moment où $\mathcal{H}$ s'apprête à sortir h_α, puis à partir de ce moment là, simule les oracles avec de nouvelles valeurs aléatoires. $\mathcal{D}_1(\omega)$ recevant la bande ω de l'étape Find, il lance $\mathcal{A}$ avec ω, lui transmet les mêmes paramètres publics, ainsi que $pk_A = (P_1, P_2)$. $\mathcal{D}_1(\omega)$ simule alors les oracles de la façon suivante, avec sa propre bande aléatoire ρ.

- **Simulation de $\mathcal{H}$:** pour toute requête d'indice $i < \alpha$, $\mathcal{D}_1(\omega)$ renvoie h_i. Sinon, il simule l'oracle aléatoire comme précédemment, mais avec de nouvelles valeurs aléatoires.

- **Simulation de Σ :** Si l'indice de la requête est $i' \leq \beta$, $\mathcal{D}_1(\omega)$ renvoie $\sigma_{i'}$. Sinon, il répond aux requêtes comme précédemment, mais avec de nouvelles valeurs aléatoires.

- **Simulation des autres oracles :** Elle se fait exactement comme dans l'étape Find.

Si $\mathcal{A}$ retourne une forge $((m', \sigma'), P'_B)$, avec $\sigma' = (Q'_1, Q'_2)$, $\mathcal{D}_1(\omega)$ appelle sa simulation de $\mathcal{H}$ pour vérifier σ' comme le faisait $\mathcal{D}_0(\omega)$ pour σ. Si σ' est invalide, ou bien se retrouve dans la trace de Σ, ou si l'indice α' de (m', Q'_1, P'_B) dans la H-List est différent de α, alors $\mathcal{D}_1(\omega)$ a perdu. L'algorithme $\mathcal{D}$ relance $\mathcal{D}_0(\omega)$ avec de nouvelles valeurs aléatoires $\rho' \neq \rho$ jusqu'à ce qu'il gagne. On note effectivement $((m', \sigma')P'_B)$, avec $\sigma' = (Q'_1, Q'_2)$ l'ultime sortie de $\mathcal{A}$.

CONCLUSION.

- **Extraction de clé**[15] **:** cette étape permet de retrouver la clé secrète du confirmeur associé à la forge de l'étape find (P'_B, m', σ'). Pour cela, $\mathcal{D}$ relance $\mathcal{D}_1$ avec une nouvelle bande aléatoire de sorte qu'en biaisant le ré-enregistrement de P'_B par $\mathcal{A}$, il puisse extraire le logarithme discret de b' de P'_B en base P. L'enregistrement étant fait par une preuve zero-knowledge de connaissance du logarithme discret de la clé à enregistrer, une modification de la bande aléatoire du protocole permettra d'extraire b'. $\mathcal{D}$ arrête $\mathcal{A}$ et mémorise b'.

- **Sortie finale :** l'algorithme $\mathcal{D}$ dispose à présent de 2 forges valides $(P_B, m, (Q_1, Q_2))$ et $(P'_B, m', (Q'_1, Q'_2))$. Les deux requêtes (m, Q_1, P_B) et (m', Q'_1, P'_B) ayant le même indice dans les $\mathsf{H\text{-}List}$ des deux simulations, il y a en fait égalité : $(m, Q_1, P_B) = (m', Q'_1, P'_B)$. Ayant extrait la clé secrète associée à la clé P'_B, $\mathcal{D}$ peut calculer les points

  $$U = x_1^{-1} b^{-1} Q_2 \text{ et } U' = x_1^{-1} b^{-1} Q'_2.$$

  $\mathcal{D}$ pose alors

  $$\begin{aligned} Q = (h_\alpha - h'_\alpha)^{-1}(U - U') &= \frac{1}{h_\alpha - h'_\alpha} \times \frac{1}{x_1 b} \times \left(\frac{r x_1 b}{x + h_\alpha} - \frac{r x_1 b}{x + h'_\alpha} \right) P \\ &= \frac{1}{(x + h_\alpha)(x + h'_\alpha)} Q_1 \end{aligned}$$

  ou bien s'arrête si $h'_\alpha = h_\alpha$. Ainsi, $U = (x + h'_\alpha)Q$ et donc

  $$U - h'_\alpha Q = xQ.$$

  Et puisque $Q_1 = (x + h_\alpha)(x + h'_\alpha)Q$, on retrouve

  $$Q_1 - (h_\alpha + h'_\alpha)xQ - h_\alpha h'_\alpha Q = x^2 Q.$$

Ainsi, $\mathcal{D}$ sort le triplet (Q, xQ, x^2Q) comme réponse à l'instance du problème FSDH.

Il reste à calculer le coût de la réduction. Il y a dans la $\mathsf{H\text{-}List}$ $q_{Tot} = q_\mathcal{H} + q_\Sigma + q_\Upsilon + q_\mathfrak{C} + q_\mathfrak{D} + 1$ hachés. Parmi ceux-ci, q_Σ ont été demandés par la simulation de l'oracle de signature Σ et, par construction, le α-ième haché dans la $\mathsf{H\text{-}List}$ ne peut pas en faire partie.

On note $\mathcal{H}_i$ l'ensemble des vecteurs $(h_1, \ldots, h_{q_{Tot}})$ qui aboutissent à une forge d'indice i, et $\varepsilon_\alpha = \Pr[(h_1, \ldots, h_{q_{Tot}}) \in \mathcal{H}_\alpha]$, la probabilité étant prise sur toutes les valeurs $h_1, \ldots, h_{q_{Tot}}$ sur $[\![1, q-1]\!]$. On a $\sum_{i=1}^{q_{Tot}} \varepsilon_i = \varepsilon$.

Lemme 3 (Splitting lemma - [199]) *Si l'on note*

$$X_\alpha = \left\{ (h_1, \ldots, h_{\alpha-1}) \in [\![1, q-1]\!]^{\alpha-1} \textit{ tels que } \Pr[(h_1, \ldots, h_{\alpha-1}, h'_\alpha, \ldots, h_{q_{Tot}}) \in \mathcal{H}_\alpha] \geq \frac{\varepsilon_\alpha}{2} \right\},$$

où la probabilité est prise sur les $h'_\alpha, \ldots, h_{q_{Tot}}$ dans $[\![1, q-1]\!]$, alors

$$\Pr[(h_1, \ldots, h_{\alpha-1}) \in X_\alpha \mid (h_1, \ldots, h_{q_{Tot}}) \in \mathcal{H}_\alpha] \geq \frac{1}{2},$$

où la probabilité est prise sur les $h_1, \ldots, h_{\alpha-1}$ dans $[\![1, q-1]\!]$.

[15] voir section 3.3.1

Ainsi, si l'on note $S^\star$ l'événement qui consiste, pour $\mathcal{D}$ à obtenir deux sous-routines gagnantes avec $\alpha = \alpha'$ et $h^0_\alpha \neq h^1_\alpha$ (les α-ièmes hachés de la première et de la seconde sous-routine sont différents). Ainsi, d'après le *splitting lemma*, la probabilité p_i que les deux sous-routines soient victorieuses au rang i et que $h^0_i \neq h^1_i$ vérifie

$$p_i \geq \frac{\varepsilon_i}{2}\left(\frac{\varepsilon_i}{2} - \frac{1}{2^k}\right).$$

Par ailleurs,

$$\begin{aligned}
\Pr[S^\star] = \sum_{i=1}^{q_{Tot}} \frac{\varepsilon_i}{2}\left(\frac{\varepsilon_i}{2} - \frac{1}{2^k}\right) &= \sum_{i=1}^{q_{Tot}} \left(\frac{\varepsilon_i}{2}\right)^2 - \frac{\varepsilon}{2^{k+1}} \\
&\geq \frac{1}{q_{Tot}}\left(\sum_{i=1}^{q_{Tot}} \frac{\varepsilon_i}{2}\right)^2 - \frac{\varepsilon}{2^{k+1}} \text{ d'après l'inégalité de Cauchy-Schwartz} \\
&\geq \frac{1}{4}\frac{\varepsilon^2}{q_{Tot}} - \frac{\varepsilon}{2^{k+1}} \\
&\geq \frac{1}{q_{Tot}}\left(\frac{\varepsilon}{2} - \frac{q_{Tot}}{2^k}\right)^2 \text{ si } \varepsilon \geq \frac{q_{Tot}}{2^{k-1}}.
\end{aligned}$$

Ce qui implique les bornes énoncées dans le théorème et conclut la preuve.

□

3.2.2.2 Invisibilité

Invisibilité faible Sous l'hypothèse xyz-DDH, le schéma DSBM reste invisible au sens faible contre une attaque à messages choisis, dans le modèle de l'oracle aléatoire. La réduction est donnée dans le théorème suivant.

Théorème 18 *Soit BDHGen un générateur de groupes BDH symétrique et soit DSBM le schéma de signatures dirigées associé. Dans le modèle de l'oracle aléatoire, pour tout adversaire weak-Inv-CMA $\mathcal{A}$ contre DSBM, de paramètre de sécurité k, ayant un avantage pour lever l'invisibilité faible $\mathbf{Adv}^{\mathsf{weak-inv-cma}}_{\mathsf{DSBM},\mathcal{A}}(k) = \varepsilon$ en temps τ, et pouvant faire $q_{\mathcal{H}}$ requêtes a l'oracle aléatoire $\mathcal{H}$, $q_{\mathfrak{C}}$ et $q_{\mathfrak{D}}$ aux oracles de confirmation et de démenti et $q_{\mathfrak{R}}$ requêtes d'enregistrement de clé, il existe un adversaire $\mathcal{D}$ contre xyz-DDH, d'avantage $\mathbf{Adv}^{\mathsf{xyz-DDH}}_{\mathsf{BDHGen},D}(k) = \varepsilon'$ en temps τ' tel que*

$$\left\{\begin{array}{l} \varepsilon' \geq \dfrac{1}{2}\varepsilon - \dfrac{q_\Sigma(q_{\mathcal{H}} + q_{\mathfrak{C}} + q_{\mathfrak{D}})}{2^k} \\ \tau' \leq q_{\mathfrak{R}}\tau + O(1). \end{array}\right.$$

Preuve :

Soit k un paramètre de sécurité, et $(q, P, \mathbb{G}, \mathbb{H}, e)$ un 5-uplet généré par $\mathsf{BDHGen}(k)$. Considérons une instance aléatoire (X, Y, Z, Q) du problème xyz-DDH, où $X = xP$, $Y = yP$ et $Z = zP$, pour $(x, y, z) \in [\![1, q_1]\!]^3$. Nous allons construire un algorithme de réduction $\mathcal{D}$ qui, en interagissant avec un attaquant weak-Inv-CMA $\mathcal{A}$ contre DSBM, va décider si $Q = xyzP$.

Soit donc $\mathcal{A}$ un attaquant weak-Inv-CMA d'avantage $\mathbf{Adv}^{\mathsf{weak-inv-cma}}_{\mathsf{DSBM},\mathcal{A}}(k) = \varepsilon$. $\mathcal{D}$, recevant $(q, P, \mathbb{G}, \mathbb{H}, e)$ et l'instance (X, Y, Z, Q) de xyz-DDH interagit avec $\mathcal{A}$ de la façon suivante. Il tire au hasard $x_2 \in [\![1, q-1]\!]$ et pose $P_1 = Y$ et $P_2 = x_2P$. $\mathcal{D}$ donne le couple (P_1, P_2) en guise de clé publique à $\mathcal{A}$, et comme clé du confirmeur challenge $pk_B = Z$.

Extraction de clé (pour un confirmeur différent du confirmeur challenge) : $\mathcal{A}$ ayant préalablement enregistré les clés $P_B = bP$ des confirmeurs de son choix, $\mathcal{D}$ retrouve la clé publique correspondante b en relançant $\mathcal{A}$ avec le même ruban aléatoire, et en ne changeant que de valeur aléatoire durant le protocole zero-knowledge pour extraire le secret. $\mathcal{D}$ peut ainsi connaître le secret de tous les confirmeurs choisis par $\mathcal{A}$, au prix d'un facteur $q_{\mathfrak{R}}$ dans le nombre d'exécutions de $\mathcal{A}$.

Simulation des oracles

– **Simulation de $\mathcal{H}$:** $\mathcal{D}$ simule $\mathcal{H}$ en tirant des valeurs au hasard dans $[\![1, q-1]\!]$ pour chaque nouvelle requête, ou renvoie la valeur déjà définie si c'est le cas, en maintenant à jour une H-List.

– **Simulation des signatures dirigées**
  - **vers le confirmeur challenge :** étant donné (m, Z), $\mathcal{D}$ tire au hasard r et h dans $[\![1, q-1]\!]$ et pose
  $$\sigma = (Q_1, Q_2) = \left(rX, \frac{r}{x_2 + h}Q\right).$$
  Si $\mathcal{H}(m, Q_1, Z)$ est déjà défini, $\mathcal{D}$ s'arrête, sinon, il rafraîchit la H-List avec h et retourne σ comme signature.
  - **vers tout autre confirmeur :** étant donné (m, P_B), $\mathcal{D}$ extrait la clé secrète b associée, puis tire au hasard r et h dans $[\![1, q-1]\!]$ et pose
  $$\sigma = (Q_1, Q_2) = \left(rP, \frac{rb}{x_2 + h}P_1\right).$$
  Si $\mathcal{H}(m, Q_1, P_B)$ est déjà défini, $\mathcal{D}$ s'arrête, sinon, il rafraichit la H-List avec h et retourne σ comme signature.

Signature challenge A l'issue de son étape Find, $\mathcal{A}$ sort un message challenge $m^\star$, dont il attend une signature. $\mathcal{D}$ tire alors au hasard $r^\star$ et $h^\star$ dans $[\![1, q-1]\!]$, puis il tire un bit $b \in \{0, 1\}$ au hasard. Il pose
$$\sigma^\star = (Q_1^\star, Q_2^\star) = \left(r^\star X, \frac{r^\star}{x_2 + h^\star}Q\right).$$

Si $b = 1$, $\mathcal{D}$ pose $\hat{m} = m^\star$, et si $b = 0$, il tire au hasard un message $\hat{m} \in \{0, 1\}^*$. Il définit alors $\mathcal{H}(\hat{m}, Q_1^\star, Z)$ comme étant $h^\star$ puis propose $\sigma^\star$ à $\mathcal{A}$ comme signature du message challenge. $\mathcal{A}$, qui a un nouvel acccès permanent aux différents oracles, les interroge à sa guise. $\mathcal{D}$ les simule exactement comme dans l'étape Find de $\mathcal{A}$. À l'issue de cette étape Guess, $\mathcal{A}$ sort un bit b'. L'algorithme $\mathcal{D}$ sort 1 si $b' = b$ et 0 sinon.

L'avantage de $\mathcal{A}$ pour trouver le bit b est essentiellement ε si $Q = xyzP$ puisqu'alors toutes les signatures simulées sont valides, et si $Q \neq xyzP$, les signatures ne sont pas valides, et $\mathcal{A}$ a alors un comportement arbitraire.

Il reste à analyser le coût de cette réduction. L'algorithme $\mathcal{D}$ s'arrête avec une probabilité $\frac{q_\Sigma(q_\mathcal{H} + q_\mathfrak{C} + q_\mathfrak{D})}{2^k}$. Ainsi, $\mathcal{D}$ résout son instance de FSDH avec probabilité au moins
$$\frac{1}{2}\mathbf{Adv}^{\mathsf{weak-inv-cma}}_{\mathsf{DSBM},\mathcal{A}}(k) - \frac{q_\Sigma(q_\mathcal{H} + q_\mathfrak{C} + q_\mathfrak{D})}{2^k}.$$

Ceci conclut la preuve.

□

Invisibilité forte L'invisibilité forte de DSBM dans le modèle de l'oracle aléatoire est reliée à un problème non standard de la famille ℓ-CAA. Il s'agit en l'occurence du problème ℓ-xyz-DCAA.

Théorème 19 *Soit BDHGen un générateur de groupes BDH symétrique et soit DSBM le schéma de signatures dirigées associé. Dans le modèle de l'oracle aléatoire, pour tout adversaire strong-Inv-CMA $\mathcal{A}$ contre DSBM, de paramètre de sécurité k, ayant un avantage pour lever l'invisibilité forte $\mathbf{Adv}^{\mathsf{strong-inv-cma}}_{\mathit{DSBM},\mathcal{A}}(k) = \varepsilon$ en temps τ, et pouvant faire $q_{\mathcal{H}}$ requêtes a l'oracle aléatoire $\mathcal{H}$, $q_{\mathfrak{C}}$ et $q_{\mathfrak{D}}$ aux oracles de confirmation et de démenti, q_{Υ} requêtes à l'oracle de conversion individuelle, $q_{\mathfrak{U}}$ requêtes à l'oracle de génération de reçu de conversion totale et $q_{\mathfrak{R}}$ requêtes d'enregistrememt de clé, il existe un adversaire $\mathcal{B}$ contre q_Σ-xyz-DCAA d'avantage $\mathbf{Adv}^{\mathsf{q_\Sigma\text{-}xyz\text{-}DCAA}}_{\mathit{BDHGen},D}(k) = \varepsilon'$ en temps τ' et un adversaire EF-CMA $\mathcal{C}$ contre DSBM tels que*

$$\begin{cases} \varepsilon' \geq \varepsilon - \mathbf{Succ}^{\mathsf{ef-cma}}_{\mathit{DSBM},\mathcal{C}}(k) \\ \tau' \leq (q_{\mathfrak{R}} + 1)\tau + O(1). \end{cases}$$

PREUVE :

Soit k un paramètre de sécurité et $(q, P, \mathbb{G}, \mathbb{H}, e)$ un 5-uplet généré par $\mathsf{BDHGen}(k)$.

On considère ici un attaquant strong-Inv-CMA $\mathcal{A}$ d'avantage $\mathbf{Adv}^{\mathsf{strong-inv-cma}}_{\mathsf{DSBM},\mathcal{A}}(k) = \varepsilon$ en temps τ pour lever l'anonymat au sens fort de DVSBM, après $q_{\mathcal{H}}$ requêtes à l'oracle de hachage, $q_{\mathfrak{C}}$ et $q_{\mathfrak{D}}$ aux oracles de confirmation et de démenti, q_{Υ} requêtes à l'oracle de conversion individuelle, $q_{\mathfrak{U}}$ requêtes à l'oracle de génération de reçu de conversion totale et pouvant enregistrer $q_{\mathfrak{R}}$ clés.

On note Bad l'événement qui consiste pour $\mathcal{A}$ à poser à l'oracle de confirmation/démenti ou de conversion une question concernant une signature valide qui n'a pas été obtenue auprès de l'oracle de signature Σ.

Soit $\Big((x_1P, x_2P, yP, zP, Q, h), (h_i, R_i, S_i)_{i\in[\![1,q_\Sigma]\!]}\Big) \in \mathbb{G}^5 \times [\![1, q-1]\!] \times \big([\![1, q-1]\!] \times \mathbb{G}^2\big)^{q_\Sigma}$ une instance aléatoire du problème q_Σ-xyz-DCAA. Nous allons décrire deux algorithmes $\mathcal{B}$ et $\mathcal{C}$ qui vont résoudre une instance de xyz-DCAA ou forger une signature selon Pr[Bad], en interagissant avec $\mathcal{A}$.

$\boxed{\text{Algorithme } \mathcal{B}}$

$\mathcal{B}$ pose $P_1 = x_1P$, $P_2 = x_2P$ et $P_B = yP$, puis initialise un compteur i à 1. Il simule alors les différents oracles de la façon suivante.

– **Simulation de $\mathcal{H}$:** pour tout message $m \in \{0,1\}^*$ proposé à l'oracle aléatoire avec le couple (Q_1, P_B), $\mathcal{B}$ renvoie la valeur $\mathcal{H}(m, Q_1, P_B)$ si celle-ci est déjà définie, sinon, il tire au hasard un élément $h \in [\![1, q-1]\!]$ qu'il renvoie comme haché en maintenant une H-List à jour.

– **Simulation de $\mathfrak{C}$ et $\mathfrak{D}$:** c'est une simulation classique de preuve d'égalité de logarithmes dans le modèle de l'oracle aléatoire si la signature a été produite par la simulation de signature de $\mathcal{B}$, et d'inégalité sinon.

– **Simulation de Σ :** étant donné $m \in \{0,1\}^*$ et la clé publique P_B d'un confirmeur, $\mathcal{B}$ tire au hasard $r \in [\![1, q-1]\!]$ et pose $Q_1 = rP$ et
  - $Q_2 = rR_i$ si $P_B = yP$ (*i.e.* le confirmeur challenge)
  - $Q_2 = rbS_i$ si $P_B = bP$ et si b a été préalablement extraite.

Si $\mathcal{H}(m, Q_1, P_B)$ a déjà été définie, et est différente de h_i, $\mathcal{B}$ tire un autre r au hasard. Sinon, il pose $\mathcal{H}(m, Q_1, P_B) = h_i$, incrémente le compteur, et sort $\sigma = (Q_1, Q_2)$ comme signature.

- **Simulation de Υ :** étant donné P_B ($\neq yP$), $m \in \{0,1\}^*$ et $\sigma = (Q_1, Q_2)$, $\mathcal{B}$ invoque la simulation de $\mathcal{H}$ sur (m, Q_1, P_B). Si σ a été obtenue lors d'une simulation de Σ alors $\mathcal{B}$ retrouve la valeur r telle que $Q_1 = rP$, et sort $\tilde{\sigma_A} = (Q_1, Q_2, rP_1)$ ou $\tilde{\sigma_B} = (Q_1, Q_2, rP_B)$. Sinon, il retourne `Invalide`.

- **Simulation de $\mathfrak{U}$:** étant donné $P_B \neq yP$, $\mathcal{B}$ calcule $I = bP_1$, où $b = \log_B(P_B)$ a été préalablement extraite.

Dans cette simulation, $\mathcal{B}$ simule parfaitement l'environnement de $\mathcal{A}$, à moins qu'à un moment dans la simulation, celui-ci demande à l'oracle $\mathfrak{C}$, $\mathfrak{D}$ ou Υ une signature (Q_1, Q_2) valide non produite par $\mathcal{B}$. Ainsi,

$$\left|\mathbf{Adv}^{\mathsf{q_\Sigma\text{-}xyz\text{-}dcaa}}_{\mathsf{BDHGen},\mathcal{B}}(k) - \mathbf{Adv}^{\mathsf{strong\text{-}inv\text{-}cma}}_{\mathsf{DVSBM},\mathcal{A}}(k)\right| \leq \Pr[\mathsf{Bad}].$$

Le temps d'exécution de $\mathcal{B}$ est au plus $(q_{\mathfrak{R}} + 1)t + O(1)$.

$\boxed{\text{Algorithme } \mathcal{C}}$

C'est essentiellement le même algorithme que pour les signatures indéniables : $\mathcal{C}$ interagit avec $\mathcal{A}$ en lui transmettant sa propre clé publique pk challenge, et simule les oracles de signature, de conversion, de confirmation et de démenti de $\mathcal{A}$ grâce à ses propres oracles. À la fin de son étape `Find`, $\mathcal{A}$ soumet à son challenger un message $m^\star$. $\mathcal{C}$ va simuler ce challenger en tirant un bit $b^\star$ au hasard et, si $b^\star = 1$ il demande à son oracle de signature une signature de $m^\star$, et si $b^\star = 0$, il tire un élément au hasard dans $\mathbb{G}^2$. On note $\sigma^\star$ l'élément qu'il a obtenu et qu'il renvoie à $\mathcal{A}$. La simulation des oracles de $\mathcal{A}$ pendant son étape `Guess` se déroule comme dans la précédente. Finalement $\mathcal{A}$ sort un bit. Durant ce jeu, $\mathcal{C}$ a classé toute les paires message/signature acceptées par l'oracle de confirmation, pour lesquelles la signature n'a pas été obtenue auprès de son oracle de signature. Finalement, $\mathcal{C}$ sort un élément quelconque de la liste, si elle n'est pas vide, et un élément aléatoire de $\{0,1\}^* \times \mathbb{G}^2$ sinon. Son temps d'exécution est celui de $\mathcal{A}$, et son succès au moins $\Pr[\mathsf{Bad}]$.

On en déduit ainsi les bornes annoncées dans le théorème.

□

3.2.3 Corollaire : une variante probabiliste de Boneh-Boyen

La sécurité du schéma de signatures dirigées DSBM est proche de celle du schéma de signatures classiques décrit dans la figure 3.6 qui peut se voir comme une variante probabiliste des signatures de Boneh-Boyen [34]. Contrairement au schéma de ces derniers, ou à celui de Zhang, Safavi-Naini, Susilo [248], l'inforgeabilité repose, dans le modèle de l'oracle aléatoire, sur un problème un peu moins exotique, à savoir le Diffie-Hellman carré flexible (FSDH). Comme pour le schéma de signatures dirigées précédent, l'utilisation de la deuxième clé permet d'obtenir un compromis intéressant dans les calculs en ligne et hors ligne qui le rend particulièrement performant. Ainsi, cette variante des signatures de Boneh et Boyen (différente de la version *Full*

Génération des paramètres publics : BBR.Setup

• k est un paramètre de sécurité,
• $(q, P, \mathbb{G}, \mathbb{H}, e)$ est généré par $\mathsf{BDHGen}(k)$
• Soit $[\{0,1\}^* \times \mathbb{G} \longrightarrow [\![1, q-1]\!]]$ une famille de fonctions de hachage h un élément pris au hasard dans cette famille.

Génération des clés : BBR.KeyGen

• $(a_1, a_2) \in [\![1, q-1]\!]^2$ est la clé privée, la clé publique est le couple (P_1, P_2), où $P_1 = a_1 P$ et $P_2 = a_2 P$

Signature : BBR.Sign

• Off-line : Alice tire au hasard des couples $(r, s) \in [\![1, q-1]\!]^2$ et calcule les points correspondants $Q_2 = \dfrac{r}{a_1 + s} P$ et $Q_1 = rP$
• On-line : pour signer le message $m \in \{0,1\}^*$, Alice tire un point Q au hasard, et calcule r' tel que $s = h(m, Q_1) + r'a_2$ et la signature de m est $\sigma = (Q_1, Q_2, r')$

Vérification : BBR.Verify

• Étant donné m, la clé publique d'Alice et la signature $\sigma = (Q_1, Q_2, r)$, Bob vérifie que

$$e(Q_2, P_1 + h(m, Q_1)P + r'P_2) = e(Q_1, P)$$

Fig. 3.6 – Signatures Boneh-Boyen probabilistes

Domain Hash probabiliste) s'insère dans les schémas de signatures de type "bas coût" comme GPS [102, 103, 202] ou GQ2 [203] par exemple.

La sécurité de ces signatures est une conséquence de l'inforgeabilité des signatures dirigées présentée précédemment :

Corollaire 3 *Sous l'hypothèse FSDH, le schéma BBR est* EF-CMA *dans le modèle de l'oracle aléatoire.*

3.3 Conclusion

Nous avons proposé dans cette section une définition des signatures dirigées totalement convertibles, puis nous avons répondu de façon positive à la question ouverte d'obtenir une conversion totate pour de telles signatures : le schéma DSBM offre ce type de conversion, tout en étant par ailleurs tout à fait compétitif en terme de performance. Une extension intéressante serait de rendre ces signatures traçables au sens de Kiayias, Tsiounis et Yung [135] : dans un contexte de signatures de groupes (voir par exemple [51]), ils donnent un moyen à des "traceurs" de lever l'anonymat d'une signature, grâce à une trappe connue du manager du groupe. Il semble être possible de modifier notre schéma pour avoir des signatures dirigées traçables. La propriété de conversion temporelle, définie pour les signatures indéniables (section 2.1.1), peut aussi être simplement obtenue grâce à l'astuce xyz.

4

Signatures à vérificateur(s) désigné(s)

Résumé : Dans cette dernière partie, nous abordons une autre famille de signatures spéciales : les signatures à vérificateurs désignés. Elles ont été introduites en 1996 par Jakobsson, Impagliazzo et Sako [125] et par Chaum [56] : ce sont en fait des preuves interactives non transférables. Nous étendons tout d'abord la notion à un groupe de vérificateurs, en développant la notion d'anonymat du signataire comme pour les signatures indéniables. Nous proposons ensuite un schéma efficace de signatures à vérificateur désigné, dont nous analysons les performances et la sécurité.

Nous proposons aussi une construction générique de signatures désignées pour un groupe de vérificateurs à partir de signatures d'anneau, qui ne protège pas l'anonymat du signataire autrement qu'en chiffrant la signature. Cependant, il peut être modifié pour donner naissance à un schéma efficace et anonyme dans le cas de 2 vérificateurs. Finalement, nous dérivons de la contruction générique basée sur les signatures d'anneau une version anonyme, au prix d'une croissance linéaire de la taille des signatures en fonction du nombre de vérificateurs.

Ces résultats ont fait l'objet de travaux, en collaboration avec Damien Vergnaud, publiés à la conférence SCN'04 [145], à la conférence ICICS'04 [144] et soumis [146].

Sommaire

4.1 Définition 132
4.1.1 Définition 133
4.1.2 Modèle de sécurité 134
4.2 Au singulier : signatures à vérificateur désigné 138
4.2.1 État de l'art 138
4.2.2 DVSBMH 145
4.2.3 Comparaison des schémas 153
4.3 Au pluriel : signatures à vérificateurs désignés 154
4.3.1 Construction générique à partir de signatures d'anneaux 155
4.3.2 Un schéma efficace dans le cas $n = 2$ 156
4.3.3 SMDVS : anonymat sans chiffrement 162
4.3.4 Conclusion 166

4.1 Définition

Les signatures à vérificateur désigné (*designated verifier signatures* - DVS) ont été introduites en 1996 par Jakobsson, Sako et Impagliazzo à Eurocrypt'96 [125] et indépendamment brevetées par Chaum la même année [56] sous le nom de *signatures privées* (*private signatures*). Historiquement, ces signatures sont liées à la notion de preuves interactives de connaissance. Jakobsson, Sako et Impagliazzo définissent des *preuves à vérificateur désigné* qui ne sont censées convaincre qu'un destinataire privilégié, choisi par le prouveur, et qui ne peut transférer sa conviction de la validité de la preuve. Pour cela, ils utilisent des *engagements à trappes*, définis dans [43] par Brassard, Chaum et Crépeau en 1988. Les signatures correspondantes, ou preuves à vérificateur désigné non-interactives, sont déduites grâce au paradigme de Fiat-Shamir [88].

La particularité de ces preuves provient du fait que le prouveur les construit en ayant choisi le vérificateur : seul ce dernier sera convaincu de leur validité, et ce essentiellement parce qu'il est capable de produire les mêmes preuves. Le vérificateur ne peut donc pas transmettre sa conviction, puisqu'il peut lui-même contrefaire cette preuve. La conception de ce type de preuve a été motivée par l'utilisation de preuves de connaissance dans des schémas de signatures, et en particulier dans les schémas de signatures indéniables. Celles-ci peuvent être sujettes à des attaques de types *blackmailing* [75,124] ou *mafia* [74], du fait que le prouveur, s'il est capable de décider *quand* il fait vérifier une signature, ne sait pas *quelle* signature il fait vérifier, ni même *à qui* (ou à combien). Dans une preuve de connaissance classique, Alice (le prouveur) prouve "cette assertion est vraie" (par exemple, "je connais le logarithme discret dans telle base de telle clé"). La version non-interactive d'une telle preuve, qu'on obtient *via* le paradigme de Fiat-Shamir et appelée *signature de connaissance*, est *transférable*, en ce sens qu'elle convainc quiconque en sa possession. L'idée des preuves à vérificateur désigné est de prouver "cette assertion est vraie OU je suis le vérificateur désigné". Ainsi, si Alice prouve à Bob "je connais le logarithme discret dans telle base de cette clé OU je suis Bob", elle va convaincre Bob qui sait bien qu'elle n'est pas Bob. Par contre, un tiers sera incapable de décider si c'est Alice ou Bob qui a produit la preuve.

La version non-interactive qui résulte de ces preuves est appelée *signature à vérificateur désigné*. Contrairement aux signatures de connaissance, celles-ci sont *non-transférables*. Elles ne vont convaincre de leur validité que le vérificateur choisi par le prouveur ou le signataire. Dans ce cas, le terme de *signatures* est bien mal choisi, puisque ces signatures brisent, outre le principe important de vérification universelle, la propriété essentielle de non-répudiation d'une signature classique. Même devant un juge, il est impossible de décider qui, du prouveur ou du vérificateur, a produit cette preuve.

Une variante particulièrement intéressante de ces signatures sont les *signatures caméléons*, introduites par H. Krawczyk et T. Rabin dans [141]. Elle sont basées sur le paradigme classique du *hash-and-sign*, et la fonction de hachage sous-jacente est en fait une fonction de hachage à trappe (fonction caméléon). Le vérificateur, qui possède la trappe de la fonction, est capable, grâce à celle-ci, de produire des forges de n'importe quel message lorsqu'il a reçu une signature. La connaissance d'une signature σ d'un message m permet au vérificateur de produire des forges de n'importe quel message $m' \neq m$. Il peut donc produire des forges indistinguables des vraies signatures produites par Alice (et donc une signature ne convainc personne d'autre *a priori*), mais Alice, et seulement elle, a la possibilité de prouver que cette signature est une forge. Alice ne peut pas répudier une signature qu'elle a produite, et peut répudier une signature qu'elle n'a pas produite.

Applications Ces signatures ont de nombreuses applications électroniques. Les signatures à vérificateur désigné peuvent servir de signature de contrat (c'est la motivation proposée dans l'article fondateur [125]), et dans l'article de Steinfeld *et al.* d'Asiacrypt'03 [231], la propriété supplémentaire d'universalité (sur laquelle nous reviendrons), est motivée par la volonté de contrôler (limiter) la dissémination des certificats. Ces signatures ont aussi été utilisées dans le cadre du vote électronique. Un des problèmes fondamentaux des protocoles électroniques de vote est de convaincre le votant que son vote a effectivement et correctement été pris en compte, tout en le protégeant contre une personne voulant connaître, acheter, ou forcer son vote. En 2000, Hirt et Sako à Eurocrypt 2000 [123] proposent un protocole de vote utilisant les signatures à vérificateur désigné, ainsi que Dall'Olio et Markowitch en 2004 dans [68]. Pointcheval, dans [195], suggère leur utilisation dans des protocoles de monnaie électronique.

La section suivante propose une définition formelle des signatures à vérificateurs désignés, qui généralise la définition de Jakobsson *et al.*, puis décrit le modèle de sécurité de ces objets. Nous étudions ensuite les signatures à vérificateur désigné, en proposant un schéma et une preuve de sa sécurité. Nous montrerons qu'il est l'un des schémas les plus performants. Finalement, nous décrirons des schémas généralisant le concept dans la dernière partie, avec une étude de leur sécurité.

4.1.1 Définition

La généralisation du concept de signatures (ou de preuves) à vérificateur désigné, au sens où l'on accepte un nombre quelconque de vérificateurs a été abordée dès 1996 par Jakobsson *et al.*, mais n'a jamais été formalisée. Nous donnons donc ici une définition précise d'un schéma de signatures à vérificateurs désignés, qui englobe naturellement la définition des signatures destinées à un seul vérificateur. Nous reviendrons sur cette dernière dans la partie 4.3. Cette définition est issue de l'article paru à ICICS'04 écrit avec Damien Vergnaud [144].

Définition 53 (Schéma de signature à n vérificateurs désignés) *Soit k et n (le nombre de vérificateurs désignés) deux entiers. Un* schéma de signature à n vérificateurs désignés n-DVS *de paramètre de sécurité k est composé de 5 algorithmes :*

- génération des paramètres publics n-DVS.Setup *: algorithme probabiliste prenant en entrée un entier k et renvoyant les* paramètres publics *;*
- génération des clés du signataire n-DVS.SKeyGen *: algorithme probabiliste prenant en entrée un entier k, les paramètres publics, et renvoyant une* paire de clés publique et secrète (pk_A, sk_A) *;*
- génération des clés des vérificateurs n-DVS.VKeyGen *: algorithme probabiliste prenant en entrée un entier k, les paramètres publics, et produisant une* paire de clés publique et secrète (pk_B, sk_B) *;*
- signature à n vérificateurs désignés n-DVS.Sign *: algorithme probabiliste ou déterministe prenant en entrée un entier k, les paramètres publics, une clé secrète sk_A, n clés publiques pk_{B_i}, $i \in [\![1, n]\!]$ et un message $m \in \{0,1\}^*$ et renvoyant σ, une* signature à $(B_1, \ldots, B_n)$-vérificateurs désignés *du message m.*
- vérification désignée n-DVS.Verify *: algorithme déterministe prenant en entrée, un entier k, les paramètres publics, une clé secrète de vérification sk_{B_i}, pour un certain $i \in [\![1, n]\!]$, une clé publique pk_A, n clés publiques pk_{B_i}, $i \in [\![1, n]\!]$, un message $m \in \{0,1\}$ et une chaîne de bits σ et testant si σ est une signature à $(B_1, \ldots, B_n)$-vérificateurs désignés valide du message m par rapport à la clé pk_A.*

Comme pour les signatures numériques classiques, on attend d'un tel schéma que toute signature correctement produite soit acceptée par l'algorithme de vérification, et ce, dans ce cas précis, pour tous les vérificateurs désignés. De même, on souhaite qu'il soit calculatoirement difficile, sans la connaissance du secret du signataire, ou de ceux de tous les vérificateurs, de produire de nouveaux couples messages/signatures valides.

Les deux propriétés supplémentaires demandées aux DVS sont des propriétés d'anonymat. La première, l'*ambiguïté du signataire*, est spécifique à ces signatures : elle correspond à l'impossibilité de savoir qui, du signataire ou du (de tous les) vérificateur(s), a produit la signature, lorsque l'on n'est pas l'un d'eux. La deuxième, appelée *anonymat du signataire*, traduit le fait qu'il est impossible de savoir qu'une signature est valide pour un signataire donné quand on n'est pas un vérificateur : face à deux signataires potentiels, un attaquant ne peut décider lequel a produit une signature donnée.

Ces propriétés sont formellement définies dans la section suivante.

4.1.2 Modèle de sécurité

Nous présentons dans cette section les définitions précises des propriétés des signatures à vérificateurs désignés, à savoir la consistance, l'inforgeabilité, le *source hiding* et l'anonymat du signataire. On note

$$n\text{-}\mathsf{DVS} = (\mathsf{Setup}, \mathsf{SKeyGen}, \mathsf{VKeyGen}, \mathsf{Sign}, \mathsf{Verify})\,.$$

4.1.2.1 Consistance

Un schéma de signatures à vérificateurs désignés doit naturellement vérifier la propriété suivante de *consistance* (*correctness*) :

Définition 54 (Consistance) *Un schéma de signature à n vérificateurs désignés est dit* consistant *si, pour tout $k \in \mathbb{N}$, pour tout* $\mathrm{params} \in \{n\text{-}\mathit{DVS.Setup}(k)\}$*, pour toute paire de clés* $(pk_A, sk_A) \in \{n\text{-}\mathit{DVS.SKeyGen}(k, \mathrm{params})\}$ *et* $\big((pk_{B_i}, sk_{B_i})\big)_{i \in [\![1,n]\!]} \in \{n\text{-}\mathit{DVS.VKeyGen}(k, \mathrm{params})\}^n$*, pour tout message $m \in \{0,1\}^*$, et pour tout $j \in [\![1,n]\!]$,*

$$n\text{-}\mathit{DVS.Verify}\big(k, \mathrm{params}, sk_{B_j}, pk_A, \{pk_{B_i}\}_{i\in[\![1,n]\!]}, m, n\text{-}\mathit{DVS.Sign}(k, \mathrm{params}, sk_A, \{pk_{B_i}\}_{i\in[\![1,n]\!]}, m)\big) = 1.$$

4.1.2.2 Inforgeabilité

Nous envisageons la sécurité contre une forge existentielle sous une attaque à messages choisis. Nous considérons donc un attaquant $\mathsf{EF\text{-}CMA}$ $\mathcal{A}$ dont la seule différence par rapport à l'adversaire d'un schéma de signatures classiques réside dans le fait qu'il a accès à un oracle de vérification Υ. En effet, il n'est pas nécessairement capable de vérifier une signature par lui-même, en particulier lorsque le schéma est anonyme (au sens de la définition 57) puisqu'alors seuls les vérificateurs désignés le peuvent. De plus, dans le cas où le nombre de vérificateurs n est strictement plus grand que 1, l'attaquant a aussi la possibilité de corrompre tous les vérificateurs désignés, sauf 1. Cela signifie que l'attaquant peut obtenir, durant son attaque, les secrets de $n-1$ vérificateurs. Dans ce cas, l'oracle de vérification n'est plus nécessaire, car l'attaquant peut vérifier les signatures en utilisant la clé secrète de vérification d'un vérificateur corrompu. L'oracle de vérification n'est donc nécessaire que pour le cas où $n = 1$.

Ainsi, l'attaque se déroule de la façon suivante : l'adversaire reçoit la clé publique du signataire et celles de tous les vérificateurs. Il interroge alors un oracle aléatoire $\mathcal{H}$, un oracle de signature

Σ, un oracle de vérification Υ, et, à n'importe quel moment pendant l'attaque, $\mathcal{A}$ interroge au plus $n-1$ fois l'oracle de "corruption" Ξ pour obtenir le secret d'un vérificateur. À l'issue de ces interactions, l'attaquant sort un couple message/signature $(m^\star, \sigma^\star)$, avec la restriction suivante : la signature $\sigma^\star$ n'a pas été obtenue auprès de l'oracle de signature (par ailleurs, le message $m^\star$ a pu être proposé à l'oracle de signature).

Formellement, nous pouvons définir la notion de forge existentielle sous une attaque à messages choisis de la façon suivante :

Définition 55 (Inforgeabilité d'un schéma de signatures à n vérificateurs désignés) *Soit k et n deux entiers et $(B_1, \ldots, B_n)$ un ensemble de n vérificateurs. Soit n-DVS un schéma de signature à n vérificateurs désignés de paramètre de sécurité k. Soit $\mathcal{A}$ un adversaire EF-CMA contre n-DVS. Considérons l'expérience aléatoire suivante :*

$$\boxed{\textit{Experiment } \mathbf{Exp}^{\mathsf{ef-cma}}_{n\text{-}\mathsf{DVS},\mathcal{A}}(k)}$$

$$\begin{array}{l}
\text{params} \leftarrow n\text{-}\mathsf{DVS.Setup}(k) \\
(pk_A, sk_A) \leftarrow n\text{-}\mathsf{DVS.SKeyGen}(k, \text{params}) \\
\textit{Pour } i = 1, \ldots, n \textit{ faire} \\
\quad (pk_{B_i}, sk_{B_i}) \leftarrow n\text{-}\mathsf{DVS.VKeyGen}(k, \text{params}) \\
(m, \sigma) \leftarrow \mathcal{A}^{\mathcal{H},\Sigma,\Upsilon,\Xi}(k, \text{params}, pk_{B_1}, \ldots, pk_{B_n}, pk_A) \\
\textit{Retourner } \bigvee_{i=1}^{n} n\text{-}\mathsf{DVS.Verify}(k, \text{params}, sk_{B_i}, pk_A, \{pk_{B_i}\}_{i \in [\![1,n]\!]}, m, \sigma)
\end{array}$$

On définit alors le succès de l'adversaire $\mathcal{A}$ comme étant

$$\mathbf{Succ}^{\textit{ef-cma}}_{n\text{-}\mathsf{DVS},\mathcal{A}}(k) = \Pr\left[\mathbf{Exp}^{\textit{ef-cma}}_{n\text{-}\mathsf{DVS},\mathcal{A}}(k) = 1\right].$$

Étant donné $\tau \in \mathbb{N}$ et ε dans $[0,1]$, le schéma n-DVS est dit (k, τ, ε)-EF-CMA sûr si aucun adversaire $\mathcal{A}$ fonctionnant en temps τ n'a un succès $\mathbf{Succ}^{\textit{ef-cma}}_{n\text{-}\mathsf{DVS},\mathcal{A}}(k) \geq \varepsilon$.

4.1.2.3 Ambiguïté du signataire

La propriété d'*ambiguïté du signataire* (*source hiding* ou *signer's ambiguity*) a été introduite pour caractériser les signatures d'anneau dans l'article d'Asiacrypt'01 de Rivest *et al.* [209], et constitue une formalisation de la propriété proposée pour les signatures à vérificateur désigné par Jakobsson *et al.*.

Intuitivement, cette ambiguité du signataire repose sur l'existence d'un algorithme Coop prenant en entrée toutes les clés secrètes des vérificateurs désignés, qui produit des chaînes de bits dont la distribution est indistinguable de la distribution de signatures produites par Alice. Cet algorithme, durant l'attaque, est simulé par un oracle Σ_B.

Un attaquant $\mathcal{A}$ lance une attaque à messages choisis au cours de laquelle, il interroge dans une première phase un oracle aléatoire, un oracle de signature Σ_A, un oracle de vérification Υ, et un oracle Σ_B. À l'issue de cette phase, l'attaquant propose à un challenger un message $m^\star$. Ce dernier est soit signé grâce à l'algorithme de signature et la clé secrète d'Alice, soit signé par les vérificateurs tous ensemble. L'attaquant doit être incapable de distinguer ces deux situations.

Définition 56 (Ambiguïté du signataire) *Soit n, k et τ trois entiers et ε un réel dans $[0,1]$. Soit n-DVS un schéma de signature à n vérificateurs désignés de paramètre de sécurité k. Soit $\mathcal{A}$ un adversaire SH-CMA contre n-DVS.*

Le schéma n-DVS est dit (k, τ, ε)-SH-CMA s'il existe un algorithme Coop fonctionnant en temps polynomial, pour lequel, en considérant l'expérience aléatoire suivante, pour un $r \in \{0,1\}$:

$\boxed{\textit{Experiment } \mathbf{Exp}^{\textsf{sh-cma}-r}_{n\text{-}\textsf{DVS},\mathcal{A}}(k)}$

params $\leftarrow n\text{-}\textsf{DVS.Setup}(k)$
$(pk_A, sk_A) \leftarrow n\text{-}\textsf{DVS.SKeyGen}(k, \text{params})$
Pour $i = 1, \ldots, n$ *faire*
  $(pk_{B_i}, sk_{B_i}) \leftarrow n\text{-}\textsf{DVS.VKeyGen}(k, \text{params})$
$(m^\star, \mathcal{I}^\star) \leftarrow \mathcal{A}^{\Sigma_A, \Sigma_B, \Upsilon, \mathcal{H}}(\textsf{find}, pk_{B_1}, \ldots, pk_{B_n}, pk_A)$
Si $r = 0$ *faire*
  $\sigma^\star \leftarrow \textsf{Coop}(k, \text{params}, m^\star, sk_{B_1}, \ldots, sk_{B_n})$
Si $r = 1$ *faire*
  $\sigma^\star \leftarrow n\text{-}\textsf{DVS.Sign}(k, \text{params}, sk_A, pk_{B_1}, \ldots, pk_{B_n}, m^\star)$
$d \leftarrow \mathcal{A}^{\Sigma_A, \Sigma_B, \Upsilon, \mathcal{H}}(\textsf{guess}, m^\star, \mathcal{I}^\star, \sigma^\star, pk_B, pk_{A_0}, pk_{A_1})$
Retourner d

et si l'on définit l'avantage de l'adversaire $\mathcal{A}$, comme étant

$$\mathbf{Adv}^{\textsf{sh-cma}}_{n\text{-}\textsf{DVS},\mathcal{A}}(k) = \left|\Pr\left[\mathbf{Exp}^{\textsf{sh-cma}-1}_{n\text{-}\textsf{DVS},\mathcal{A}}(k) = 1\right] - \Pr\left[\mathbf{Exp}^{\textsf{sh-cma}-0}_{n\text{-}\textsf{DVS},\mathcal{A}}(k) = 1\right]\right|,$$

aucun adversaire $\mathcal{A}$ fonctionnant en temps τ n'a un avantage $\mathbf{Adv}^{\textsf{sh-cma}}_{n\text{-}\textsf{DVS},\mathcal{A}}(k) \geq \varepsilon$.

Dans les schémas que nous proposerons par la suite, nous exigerons que cette propriété soit satisfaite de manière inconditionnelle, c'est à dire que l'attaquant a exactement 1 chance sur 2 de réussir son attaque. Il suffira alors d'exhiber l'algorithme Coop de coopération entre les vérificateurs, qui produit des chaînes de bits dont la distribution est la même que celle produite par Alice lorsqu'elle signe.

4.1.2.4 Anonymat du signataire

Une notion supplémentaire d'anonymat est parfois nécessaire pour s'assurer qu'aucune oreille indiscrète ne puisse identifier un signataire. Nous formalisons cette notion d'anonymat du signataire (*privacy of signer's identity*), qui est inspirée de la notion de *key-privacy* pour le chiffrement proposée à Asiacrypt'01 par Bellare, Boldyreva, Desai et Pointcheval [14]. Celle-ci modélise le fait qu'on ne peut pas déterminer grâce à quelle clé un chiffré donné a été produit (voir la partie 3.1.2.1). Cette notion d'anonymat existe aussi pour les signatures indéniables comme nous l'avons vu dans la partie 2.1.2.3 : elle est étudiée par Galbraith et Mao dans leur article de CT-RSA'03 [95]. On pourrait d'ailleurs, pour les signatures à vérificateurs désignés, étudier la propriété d'invisibilité des signatures indéniables, ces 2 notions étant essentiellement équivalentes.

Définition Le scénario de l'attaque est le suivant. On considère un adversaire PSI-CMA $\mathcal{A}$ qui fonctionne en 2 étapes. Dans la première étape guess, l'attaquant reçoit deux clés publiques pk_{A_0} et pk_{A_1} de deux signataires potentiels, et les clés publiques des vérificateurs désignés $pk_{B_1}, \ldots, pk_{B_n}$. À l'issue d'un certain nombre de requêtes avec un oracle aléatoire $\mathcal{H}$, deux oracles de signatures Σ_{A_0} et Σ_{A_1} correspondant aux clés pk_{A_0} et pk_{A_1}, et un oracle de vérification Υ, l'adversaire sort un message $m^\star$ (et de l'information $\mathcal{I}^\star$ d'état). Dans l'étape suivante find, l'attaquant reçoit une signature à $(B_1, \ldots, B_n)$-vérificateurs désignés challenge $\sigma^\star$ qui a été produite grâce à l'une des deux clés secrètes sk_{A_0} ou sk_{A_1} (tirée au hasard) sur le message $m^\star$. Ayant à nouveau accès aux oracles, il doit tenter de déterminer avec quelle clé correspondante à pk_{A_0} ou pk_{A_1} a été utilisée. La seule restriction imposée à l'attaquant est de ne pas soumettre le couple $(m^\star, \sigma^\star)$ à l'oracle de vérification dans la deuxième étape.

Définition 57 (Anonymat du signataire) *Soit k et n deux entiers et $(B_1, \ldots, B_n)$ un ensemble de n entités. Soit n-DVS un schéma de signature à n vérificateurs désignés de paramètre de sécurité k. Soit $\mathcal{A}$ un adversaire PSI-CMA contre n-DVS. Considérons les expériences aléatoires suivantes pour un $r \in \{0, 1\}$:*

$$\boxed{\textit{Experiment } \mathbf{Exp}^{\textsf{psi-cma}-r}_{n\textsf{-DVS},\mathcal{A}}(k)}$$

$$\begin{array}{l}
\text{params} \leftarrow n\textsf{-DVS.Setup}(k) \\
(pk_{A_0}, sk_{A_0}) \leftarrow n\textsf{-DVS.SKeyGen}(k, \text{params}) \\
(pk_{A_1}, sk_{A_1}) \leftarrow n\textsf{-DVS.SKeyGen}(k, \text{params}) \\
\textit{Pour } i = 1, \ldots, n \textit{ faire} \\
\quad (pk_{B_i}, sk_{B_i}) \leftarrow n\textsf{-DVS.VKeyGen}(k, \text{params}) \\
(m^\star, \mathcal{I}^\star) \leftarrow \mathcal{A}^{\Sigma_{A_0}, \Sigma_{A_1}, \Upsilon, \mathcal{H}}(\textsf{find}, pk_{B_1}, \ldots, pk_{B_n}, pk_{A_0}, pk_{A_1}) \\
\sigma^\star \leftarrow n\textsf{-DVS.Sign}(k, \text{params}, sk_{A_r}, pk_{B_1}, \ldots, pk_{B_n}, m^\star) \\
d \leftarrow \mathcal{A}^{\Sigma_{A_0}, \Sigma_{A_1}, \Upsilon, \mathcal{H}}(\textsf{guess}, m^\star, \mathcal{I}^\star, \sigma^\star, pk_B, pk_{A_0}, pk_{A_1}) \\
\textit{Retourner } d
\end{array}$$

*On définit l'*avantage *de l'adversaire $\mathcal{A}$, comme étant*

$$\mathbf{Adv}^{\textsf{psi-cma}}_{n\textsf{-DVS},\mathcal{A}}(k) = \left| \Pr\left[\mathbf{Exp}^{\textsf{psi-cma}-1}_{n\textsf{-DVS},\mathcal{A}}(k) = 1\right] - \Pr\left[\mathbf{Exp}^{\textsf{psi-cma}-0}_{n\textsf{-DVS},\mathcal{A}}(k) = 1\right] \right|.$$

Étant donné un entier τ et ε un réel dans $[0, 1]$, le schéma n-DVS est dit (k, τ, ε)-PSI-CMA sûr si aucun adversaire $\mathcal{A}$ fonctionnant en temps τ n'a un avantage $\mathbf{Adv}^{\textsf{psi-cma}}_{n\textsf{-DVS},\mathcal{A}}(k) \geq \varepsilon$.

Chiffrement IND-CCA2 implique anonymat Il existe une manière simple, mais pas satisfaisante d'obtenir la propriété d'anonymat du signataire, qui consiste à chiffrer convenablement la signature. Cette solution n'est pas satisfaisante à cause du surcoût lié à l'utilisation d'un algorithme de chiffrement, en particulier dans un contexte multi-utilisateurs où il est nécessaire d'échanger une clé de chiffrement. Jakobsson *et al.* écrivaient qu'il était suffisant de chiffrer de façon probabiliste la signature pour avoir l'anonymat du signataire. Plus précisément, nous montrons dans la proposition suivante que le chiffrement par un algorithme IND-CCA2 de la signature permet d'obtenir la propriété d'anonymat du signataire. On peut faire l'hypothèse que les vérificateurs désignés partagent une clé de déchiffrement, aussi, ne considérons-nous qu'un seul vérificateur dans la proposition.

Proposition 7 *Soit k un entier, et soit DVS un schéma de signature à vérificateur désigné de paramètre de sécurité k. Soit $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ un schéma de chiffrement à clé publique sémantiquement sûr face à une attaque à chiffrés choisis adaptative (IND-CCA2) de paramètre de sécurité k.*

Alors pour tout entier k et pour tout adversaire PSI-CMA $\mathcal{A}$ ayant un avantage $\mathbf{Adv}^{\textsf{psi-cma}}_{\textsf{DVS}^{\Pi},\mathcal{A}}(k)$ en temps t, après avoir fait $q_{\mathcal{H}}$, q_{Σ} et q_{Υ} requêtes à l'oracle aléatoire, de signature et de vérification, il existe un adversaire IND-CCA2 $\mathcal{A}'$ contre Π, adressant $q_{\mathcal{H}}$ requêtes à l'oracle aléatoire et q_{Υ} requêtes à l'oracle de déchiffrement, en temps t et d'avantage

$$\mathbf{Adv}^{\textsf{ind-cca2}}_{\Pi,\mathcal{A}'}(k) = \mathbf{Adv}^{\textsf{psi-cma}}_{\textsf{DVS}^{\Pi},\mathcal{A}}(k).$$

Preuve : On considère qu'une signature obtenue par DVS.Sign est chiffrée avec l'algorithme $\mathcal{E}$ de Π, avec la clé publique $E_{pk_B} \in \mathcal{K}(k)$ du vérificateur désigné. On suppose l'existence d'un adversaire PSI-CMA $\mathcal{A}$ d'avantage $\mathbf{Adv}^{\mathsf{psi-cma}}_{\mathsf{DVS}^{\Pi},\mathcal{A}}(k)$. On construit l'algorithme $\mathcal{A}'$ contre la propriété IND-CCA2 de la façon suivante : il prend en entrée la clé publique E_{pk_B} et choisit deux paires de clés de signature (sk_{A_0}, pk_{A_0}) et (sk_{A_1}, pk_{A_1}), ainsi qu'une paire de clés de vérification (sk_B, pk_B). $\mathcal{A}$ reçoit les clés publiques E_{pk_B}, pk_{A_0}, pk_{A_1} et pk_B. À chaque requête de signature de la part de $\mathcal{A}$ (et ce dans ses différentes phases guess et find), $\mathcal{A}'$ répond grâce à sa connaissance des clés secrètes sk_{A_0} et sk_{A_1}. De même, $\mathcal{A}'$ répond aux requêtes de vérification grâce à la clé sk_B et à l'oracle de déchiffrement. Lorsque l'étape guess de $\mathcal{A}$ se termine, il exhibe un message $m^\star \in \{0,1\}^*$. $\mathcal{A}'$ calcule alors deux signatures $\sigma_0^\star$ et $\sigma_1^\star$ du message $m^\star$ à l'aide de l'algorithme DVS.Sign et des deux clés sk_{A_0} et sk_{A_1}. Le challenger IND-CCA2 reçoit ses deux signatures, tire un bit $b \in \{0,1\}$ au hasard et renvoie le chiffré de $\sigma_b^\star$. Celui-ci est alors donné comme challenge PSI-CMA à $\mathcal{A}$. Ce dernier, après s'être entretenu avec les oracles, sort un bit b' à la fin de son étage find. Par définition de $\mathcal{A}$, $b = b'$ avec une probabilité de $\mathbf{Adv}^{\mathsf{psi-cma}}_{\mathsf{DVS}^{\Pi},\mathcal{A}}(k)$. Ainsi, $\mathcal{A}'$ distingue correctement les deux messages chiffrés $\sigma_0^\star$ et $\sigma_1^\star$ avec le même avantage

$$\mathbf{Adv}^{\mathsf{ind-cca2}}_{\Pi,\mathcal{A}'}(k) = \mathbf{Adv}^{\mathsf{psi-cma}}_{\mathsf{DVS}^{\Pi},\mathcal{A}}(k).$$

□

4.2 Au singulier : signatures à vérificateur désigné

Dans cette partie, nous nous intéressons aux signatures à vérificateur désigné originelles, et après un aperçu des constructions existantes, nous proposons un schéma basé sur les applications bilinéaires, y compris les applications "bas-coût", et analysons sa sécurité et ses performances.

4.2.1 État de l'art

Dans leur article fondateur du concept de preuve à vérificateur désigné, Jakobsson *et al.* [125] proposent une construction de preuves à vérificateur désigné interactive pour des signatures indéniables, en utilisant le concept d'engagement à trappe introduit en 1988 par Brassard, Chaum et Crépeau dans [43]. Ils partent du schéma de confirmation des signatures indéniables de Chaum [54] et le rendent à vérificateur désigné à partir de n'importe quel engagement à trappe. La problématique naturelle de rendre ces preuves non-interactives est ensuite abordée : ils utilisent simplement le paradigme de Fiat-Shamir [88] pour transformer un protocole d'identification en trois passes en un protocole en une passe. Le schéma proposé dans cet article a été récemment cryptanalysé par Saeednia, Kremer et Markowitch [211] et par Wang [241] : un attaquant peut déduire d'une paire valide message/signature à destination du vérificateur, un couple message choisi/fausse signature qui sera accepté par le vérificateur. Il suffit, pour le réparer, d'utiliser une fonction de hachage pour relier le message et certains aléas. Ce premier schéma ne possède pas la propriété d'anonymat du signataire.

Comme suggéré par Rivest, Shamir et Tauman à Asiacrypt'01 dans [209], un schéma de signature d'anneau réduit à 2 utilisateurs est un schéma de signature à vérificateur désigné, qui n'offre pas non plus la propriété d'anonymat du signataire. La propriété d'ambiguïté du signataire est justement une propriété des signatures d'anneau.

Dans leur article à ICISC'03 [211], Saeednia, Kremer et Markowitch proposent un schéma basé sur les signatures de Schnorr. Leur schéma est très efficace, protège l'anonymat, mais la sécurité n'est étudiée que dans un modèle faible. Nous montrons dans la suite que le schéma atteint toutes les propriétés des signatures à vérificateurs désignés dans le modèle présenté précédemment.

Un schéma de DVS basé sur l'identité a été proposé par Susilo, Zhang et Mu [234], qui peut se voir comme une variante basée sur l'identité du schéma de Saeednia *et al.*, à partir d'applications bilinéaires.

À Asiacrypt'03 [231], Steinfeld, Bull, Wang et Pieprzyk proposent un schéma utilisant des applications bilinéaires. Ils introduisent une notion particulièrement intéressante de signatures à vérificateur désigné *universelles*, dont nous parlerons plus en détails dans la section suivante. Steinfeld, Wang et Pieprzyk ont aussi proposé, à PKC 2004 [232], des variantes des signatures RSA et Schnorr universelles à vérificateur désigné. Les schémas sont nommés $\mathsf{SchUDVS}_1$, $\mathsf{SchUDVS}_2$ et $\mathsf{RSAUDVS}$. Une comparaison des performances de certains de ces schémas est faite dans la partie 4.2.3.

Remarques sur la sécurité de certains de ces schémas La sécurité des signatures à vérificateur désigné n'a été véritablement formalisée que récemment : dans [232] par Steinfeld *et al.* propose un modèle formel qui améliore celui d'Asiacrypt'03 [231], puisque l'attaquant a la possibilité d'accéder à un oracle de vérification (ce n'était pas le cas dans [231]), et peut obtenir aussi des signatures destinées à d'autres vérificateurs.

Aussi, dans [211], Saeednia *et al.* ne prouvent la sécurité de leur schéma (appelons le SKM) que contre des attaques sans message. Ce n'est pas satisfaisant pour valider un schéma, mais en fait, en utilisant la technique de preuve du théorème 23, on peut montrer :

Théorème 20 (Unforgeabilité du schéma de SKM) *Soit k un entier et $\mathcal{A}$ un attaquant EF-CMA contre le schéma de SKM avec paramètre de sécurité k, dans le modèle de l'oracle aléatoire, qui produit une forge existentielle avec succès $\varepsilon = \mathbf{Succ}^{\mathsf{ef-cma}}_{SKM,\mathcal{A}}(k)$, en temps τ, après $q_{\mathcal{H}}$, q_Σ et q_Υ requêtes aux oracles de hachage, de signature et de vérification.*

Alors il existe $\varepsilon' \in [0,1]$ et $\tau' \in \mathbb{N}$ vérifiant

$$\begin{cases} \varepsilon' \geq \varepsilon - \dfrac{(q_{\mathcal{H}} + q_\Sigma)q_\Sigma + q_\Upsilon}{2^k}, \\ \tau' \leq \tau + (q_\Sigma + q_\Upsilon)\left(2T_{Exp-\mathbb{G}} + O(1)\right) + (q_{\mathcal{H}} + q_\Sigma)(q_{\mathcal{H}} + q_\Sigma + q_\Upsilon)\left(T_{DDH} + O(1)\right), \end{cases}$$

et tels que le problème Gap Diffie-Hellman (GDH) peut être résolu avec un probabilité ε', en temps τ'.

La notion d'anonymat du signataire n'a pas été systématiquement étudiés dans les schémas existants. En effet, le cadre pour cette étude a été fourni dans l'article avec Vergnaud de SCN'04 [145], et nous avons montré les résultats suivants concernant les schémas proposés par Saeednia *et al.* [211] et par Steinfeld *et al.* dans [231] et [232] :

Théorème 21 (Anonymat du schéma de SKM) *Soit k en entier et $\mathcal{A}$ un adversaire PSI-CMA, dans le modèle de l'oracle aléatoire, contre SKM, de paramètre de sécurité k, qui lève l'anonymat du signataire avec un avantage $\varepsilon = \mathbf{Adv}^{\mathsf{psi-cma}}_{SKM,\mathcal{A}}(k)$, et temps τ, après $q_{\mathcal{H}}$, q_Σ, q_Υ requêtes aux oracles aléatoires, de signature et de vérification.*

Alors il existe $\varepsilon' \in [0,1]$ et $\tau' \in \mathbb{N}$ vérifiant

$$\begin{cases} \varepsilon' \geq \dfrac{\varepsilon}{2} - \dfrac{(q_{\mathcal{H}} + q_\Sigma)q_\Sigma + q_\Upsilon}{2^k} \\ \tau' \leq \tau + (q_\Sigma + q_\Upsilon)\left(2T_{Exp-\mathbb{G}} + O(1)\right) + (q_{\mathcal{H}} + q_\Sigma)(q_{\mathcal{H}} + q_\Sigma + q_\Upsilon)\left(T_{DDH} + O(1)\right) \end{cases}$$

et tels que le problème Gap Diffie-Hellman (GDH) peut être résolu avec un probabilité ε', en temps τ'.

Théorème 22 (Anonymat de $\mathsf{SchUDVS}_1$) *Soit k un entier et $\mathcal{A}$ un adversaire PSI-CMA, dans le modèle de l'oracle aléatoire, contre $\mathit{SchUDVS}_1$ avec paramètre de sécurité k, d'avantage $\varepsilon = \mathbf{Adv}^{\mathsf{psi-cma}}_{\mathit{SchUDVS}_1,\mathcal{A}}(k)$ pour lever l'anonymat du signataire, en temps τ, après $q_{\mathcal{H}}$, q_{Σ}, q_{Υ} requêtes aux oracles aléatoire, de signature et de vérification.*

Alors il existe $\varepsilon' \in [0,1]$ et $\tau' \in \mathbb{N}$ vérifiant

$$\begin{cases} \varepsilon' \geq \dfrac{\varepsilon}{2} - \dfrac{q_{\Upsilon} + q_{\mathcal{H}} q_{\Sigma} + 1}{2^k} \\ \tau' \leq \tau + (q_{\Sigma} + q_{\Upsilon})(3T_{Exp-\mathbb{G}} + O(1)) \end{cases}$$

tels que le problème décisionnel Diffie-Hellman (DDH) peut être résolu avec une probabilité ε' en temps τ'.

Les preuves de ces théorèmes sont des adaptations des preuves des théorèmes sur le schéma que nous avons proposé à SCN'04, ou de celles de Bellare *et al.* sur la *key-privacy* [14].

4.2.1.1 Signatures à vérificateur désigné universelles

À Asiacrypt'03 [231], R. Steinfeld, L. Bull, H. Wang et J. Pieprzyk introduisent le concept de signatures à vérificateur désigné *universelles*, en considérant la possibilité de *désigner* un vérificateur privilégié comme une propriété additionnelle d'un schéma de signature classique. L'idée est que toute personne en possession d'une signature (qu'elle n'a pas nécessairement émise elle-même) peut modifier cette signature, *désigner* un vérificateur qui, alors, sera le seul à pouvoir être convaincu par sa validité, et ne pourra pas transmettre sa conviction. Dans le cadre d'une infrastructure à clés publiques où les clés des utilisateurs sont certifiées, il est parfois préférable, pour protéger les données contenues dans les certificats, d'éviter une dissémination de ceux-ci. Ainsi, les signatures à vérificateur désigné universelles permettent de contrôler la vérification d'un certificat (qui est une signature émise par une autorité, et qui pourra être modifiée par le détenteur du certificat, pour contrôler les vérificateurs).

Un schéma de signatures à vérificateur désigné universelles est alors composé de 7 algorithmes, dont les algorithmes classiques de signatures :

1. génération des paramètres publics GC
2. génération des clés du signataire GKC
3. signature S
4. vérification publique V

et des algorithmes supplémentaires de désignation :

5. génération des clés du vérificateur GKV
6. désignation CDV
7. vérification désignée VDV
8. enregistrement de la clé d'un signataire P_{KR}

Ce dernier algorithme d'enregistrement sert à s'assurer que le vérificateur connaît effectivement la clé secrète associée à la clé de vérification, et ce pour garantir la non-transférabilité de la vérification. Dans leur modèle de sécurité présenté dans [231], Steinfeld *et al.* ne permettent pas à l'attaquant, dans sa tentative de forge ("DV-Unforgeability") d'interroger un oracle de vérification.

Génération des paramètres publics : DVSBM.GC = BLS.Setup

Génération des clés du signataire : DVSBM.GKC = BLS.KeyGen $\rightsquigarrow (a, P_A)$

Génération des clés du vérificateur : DVSBM.GKV = BLS.KeyGen $\rightsquigarrow (b, P_B)$

Signature : DVSBMH.S = BLS.Sign $\rightsquigarrow (m, \sigma)$

Verification publique : DVSBM.V = BLS.Verify

Désignation : DVSBM.CDV

- Étant donné P_A, P_B et (m, σ) la signature désignée est $\hat{\sigma} = e(P_B, \sigma)$.

Vérification désignée : DVSBM.VDV

- Étant donné P_A et b la signature est acceptée si et seulement si $\hat{\sigma} = e(P_A, bH(m))$

Fig. 4.1 – DVSBM : description

La figure 4.1 décrit le schéma DVSBM de [231] qui est basé sur les signatures courtes BLS, et qui permet donc de transformer toute signature de ce type en une signature à vérificateur désigné.

Dans [231] Steinfeld *et al.* ont apporté aux signatures courtes de Boneh *et al.* [38] le caractère universel des signatures, et dans [232], Steinfeld, Wang et Pieprzyk élaborent des versions basées sur les signatures de Schnorr [215] et sur des signatures RSA. Dans aucun des deux cas, les auteurs n'étudient la propriété d'anonymat du signataire. Nous avons montré dans la section précédente que le schéma $\mathsf{SchUDVS}_1$ de [232] était anonyme. Les signatures de DVSBM sont déterministes, et ne peuvent donc pas être anonymes. Dans leur article d'Asiacrypt'03, Steinfeld *et al.* proposent également une équivalence entre une certaine classe de signatures incluant les signatures à vérificateur désigné universelles *déterministes* et une classe de chiffrements basés sur l'identité, dans laquelle se trouve le chiffrement de Boneh et Franklin [35, 36].

4.2.1.2 MAC et DVS

Un MAC (pour *Message Authentication Code*) est un algorithme qui renvoie une chaîne de bits $\tau \in \{0,1\}^\ell$ calculée à partir d'un message $m \in \{0,1\}^*$ et d'une clé secrète. La vérification d'un MAC se fait en le recalculant, grâce au partage de la clé secrète. Ce mécanisme permet de s'assurer qu'un message reçu n'a pas été modifié.

Dans [207], Rivest propose un nouveau moyen d'apporter la confidentialité dans une communication, littéralement en "rajoutant de l'ivraie au bon grain" (*chaffing and winnowing*). Sa technique consiste à rajouter à chaque paquet d'information (en clair et numéroté) un MAC calculé avec une clé partagée avec l'interlocuteur : c'est le bon grain. Puis il faut rajouter l'ivraie en noyant les paquets d'information pertinente dans des paquets de fausses informations accompagnées de MAC invalides. Un exemple de communication pourrait être[16] :

- (1,Hi Larry,532105)
- (1,Hi Bob,465231)
- (2,Meet me at,782290)
- (2,I'll call you at,793122)
- (3,6PM,891231)
- (3,7PM,344287)
- (4,Yours-Susan,553419)
- (4,Love-Alice,312265)

où, parmi les deux messages potentiels reçus, un seul est accompagné d'un MAC valide. Un attaquant, ne connaissant pas la clé secrète, ne peut pas distinguer un bon paquet (le bon grain) d'un mauvais (l'ivraie). En décrivant cette méthode (certes inefficace[17]), Rivest a suggéré l'utilisation des signatures à vérificateur désigné de Jakobsson *et al.* [125]. Comme lui a fait remarquer Senderek dans [222], ces signatures n'assurent pas leur rôle d'ivraie, puisqu'un utilisateur quelconque peut "vérifier" la signature en ce sens qu'il saura qu'elle vient d'Alice ou de Bob.

Par ailleurs, les signatures à vérificateur désigné ayant la propriété d'anonymat du signataire introduite dans la définition 57 peuvent remplacer les MAC dans le schéma de Rivest, puisque dans ce cas, elles apparaîssent aux yeux d'un utilisateur quelconque comme de l'aléa.

Bien que l'étude des signatures à vérificateurs désignés se soit faite dans un contexte de cryptographie à clé publique. l'exemple précédent suggère qu'un outil classique de la cryptographie à clé secrète est proche de ces signatures : les MAC ou *Message Authentication Code*.

Définition La définition formelle d'un MAC est la suivante :

Définition 58 (Schéma d'authentification de message) *Soit k et ℓ deux entiers, un schéma d'authentification de message SM de paramètre de sécurité k est composé de 3 algorithmes :*
- $\mathsf{SM.KeyGen}$ *: algorithme probabiliste prenant en entrée un paramètre de sécurité k et retournant une clé $\mathcal{K}$,*
- $\mathsf{SM.MAC}$ *: algorithme déterministe ou probabiliste prenant en entrée un message $m \in \{0,1\}^*$ et une clé $\mathcal{K}$ et retournant soit une chaîne de bits $\tau \in \{0,1\}^\ell$ de taille fixe, appelée elle aussi MAC, soit le symbole $\perp$ pour indiquer l'echec du calcul,*
- $\mathsf{SM.Verify}$ *: algorithme déterministe prenant en entrée un message $m \in \{0,1\}^*$, un MAC τ et une clé $\mathcal{K}$ et retournant un bit valant 1 si τ est un MAC valide pour m, 0 sinon.*

Si la génération de MAC est déterministe, l'algorithme de vérification est l'algorithme MAC accompagné d'un test d'égalité.

Les propriétés de sécurité d'un schéma d'authentification de message SM (qu'on peut trouver dans [159]) sont l'inforgeabilité et l'indistinguabilité.

[16] cet exemple est tiré de [207]

[17] à noter que la confidentialité peut être rajoutée sans être souhaitée par Alice ou Bob, puisque si leur communication est mélangée à une communication entre David et Elaine, les paquets de ces derniers jouent le rôle de l'ivraie. La confidentialité est alors assurée sans aucune clé de chiffrement

Inforgeabilité L'attaquant a la possibilité d'interroger un oracle de génération de MAC $\mathcal{O}_{\mathcal{M}}$ et, lorsque celui-ci est probabiliste, un oracle de vérification $\mathcal{O}_{\mathcal{V}}$. Son but est de produire un nouveau MAC soit pour un message pour lequel aucun MAC n'a été demandé à l'oracle $\mathcal{O}_{\mathcal{M}}$, et dans ce cas on parle d'inforgeabilité faible (*weak unforgeability*), soit pour n'importe quel message (*i.e.* même en ayant connaissance de plusieurs MAC sur ce message) : c'est l'inforgeabilité forte (*strong unforgeability*).

Définition 59 (Inforgeabilité (forte) d'un schéma de MAC) *Soit k un entier et SM un schéma de MAC de paramètre de sécurité k. Soit $\mathcal{A}$ un adversaire SUF-CMA contre SM. Considérons l'expérience aléatoire suivante :*

$$\boxed{\textit{Experiment } \mathbf{Exp}^{\mathsf{suf-cma}}_{\mathsf{SM},\mathcal{A}}(k)}$$
$$\begin{array}{l} \mathcal{K} \leftarrow \mathsf{SM.KeyGen}(k) \\ (m,\tau) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathcal{M}},\mathcal{O}_{\mathcal{V}}}(k) \\ \textit{Retourner } \mathsf{SM.Verify}(m,\tau,\mathcal{K}). \end{array}$$

On définit le succès de l'adversaire $\mathcal{A}$ comme étant

$$\mathbf{Succ}^{\textsf{suf-cma}}_{\mathrm{MAC},\mathcal{A}}(k) = \Pr\left[\mathbf{Exp}^{\textsf{suf-cma}}_{\mathsf{SM},\mathcal{A}}(k) = 1\right].$$

Soit τ un entier et ε un réel dans $[0,1]$. Le schéma MAC est dit (k,t,ε)-SUF-CMA sûr si aucun adversaire SUF-CMA $\mathcal{A}$ fonctionnant en temps t n'a un succès $\mathbf{Succ}^{\textsf{suf-cma}}_{\mathrm{MAC},\mathcal{A}}(k) \geq \varepsilon$.

On peut se référer à l'article de Bellare, Kilian et Rogaway [17, 18] pour des CBC-MAC déterministes, ainsi qu'à Bellare et Namprempre [20].

Indistinguabilité Pour mesurer l'indistinguabilité d'un schéma de MAC, on le compare à une fonction pseudo-aléatoire (PRF). La définition formelle est la suivante.

Définition 60 (Indistinguabilité d'un MAC) *Soit k un entier et SM un schéma de MAC de paramètre de sécurité k. Soit $\mathcal{A}$ un adversaire PRF contre SM. Considérons les expériences aléatoires suivantes pour un $r \in \{0,1\}$:*

$$\boxed{\textit{Experiment } \mathbf{Exp}^{\textsf{prf}-r}_{\mathsf{SM},\mathcal{A}}(k)}$$
$$\begin{array}{l} \textit{Si } r = 0 \textit{ faire} \\ \quad \mathcal{K} \leftarrow \mathsf{SM.KeyGen}(k) \textit{ et } f = \mathsf{SM.MAC} \\ \textit{sinon } f \xleftarrow{R} \{\{0,1\}^* \longrightarrow \{0,1\}^{\ell}\} \\ d \leftarrow \mathcal{A}^{\mathcal{O}_f}(k) \\ \textit{Retourner } d \end{array}$$

*On définit l'*avantage *de l'adversaire $\mathcal{A}$, comme étant*

$$\mathbf{Adv}^{\mathsf{prf}}_{\mathsf{SM},\mathcal{A}}(k) = \left|\Pr\left[\mathbf{Exp}^{\mathsf{prf}-1}_{\mathsf{SM},\mathcal{A}}(k) = 1\right] - \Pr\left[\mathbf{Exp}^{\mathsf{prf}-0}_{\mathsf{SM},\mathcal{A}}(k) = 1\right]\right|.$$

Soit τ un entier et ε un réel dans $[0,1]$. Le schéma SM est dit (k,τ,ε)-PRF indistinguable si aucun adversaire PRF $\mathcal{A}$ fonctionnant en temps τ n'a un avantage $\mathbf{Adv}^{\textsf{prf}}_{\mathsf{SM},\mathcal{A}}(k) \geq \varepsilon$.

Bellare, Kilian et Rogaway ont montré dans [17] que si un schéma de MAC était indistinguable, alors il était inforgeable. La définition pour des MAC probabilistes se déduit de la définition précédente.

Au vue des propriétés satisfaites par les MAC, ils sont des candidats de choix comme signatures à vérificateur désigné unique. La notion d'ambiguïté du signataire est par ailleurs satisfaite naturellement, puisque, par définition, Alice et Bob peuvent produire le même MAC. Les algorithmes de génération de MAC sont généralement performants bien que leur utilisation nécessite un échange préalable d'une clé.

4.2.1.3 Forge universelle sur le schéma de DVS de Sehkar

Récemment, Sekhar a proposé dans International Journal of Computer Mathematics [220] un schéma de signature avec recouvrement de message, qu'il affirme être un schéma à vérificateur désigné. Il prétend aussi que ce schéma protège l'identité du signataire. Il s'agit en fait d'un schéma de signature à confirmeur désigné, défini par Chaum dans [55]. Avec Jacques Traoré et Damien Vergnaud, nous avons montré dans l'*International Journal of Computer Mathematics* que l'on peut facilement forger une signature de tout message, même ne connaissant au préalable aucune signature [143].

Description du schéma Le schéma de Sekhar fonctionne de la façon suivante. Les entiers p et q sont deux grands premiers avec $q \mid p-1$, et α est un élément de $(\mathbb{Z}/p\mathbb{Z})^*$ d'ordre q.

Alice possède un couple de clés (x_A, y_A), où $x_A \in (\mathbb{Z}/q\mathbb{Z})^*$ est sa clé secrète et $y_A = \alpha^{x_A}$ est sa clé publique.

Bob a aussi sa paire de clés secrète/publique (x_B, y_B) avec $x_B \in (\mathbb{Z}/q\mathbb{Z})^*$ et $y_B = \alpha^{x_B}$.
Pour signer $m \in (\mathbb{Z}/p\mathbb{Z})^*$, Alice procède de la façon suivante :

1. elle tire au hasard $(k_1, k_2) \in (\mathbb{Z}/q\mathbb{Z})^*$ et calcule $U = \alpha^{k_1 - k_2} \mod p$;
2. elle calcule ensuite $r = y_B^{k_1} U m \mod p$ puis
3. $s = k_2 x_A^{-1} - r \mod q$.

La signature est le triplet $\sigma = (r, s, U)$.
Une fois en possession de la signature, Bob retrouve le message en calculant

$$m = r \left(y_A^{(r+s)} U \right)^{-x_B} U^{-1} \mod p.$$

Bob peut alors prouver que le triplet (r, S, U) est une signature d'Alice valide, en exécutant un protocole zéro-knowledge classique avec une tierce partie.

Forge universelle par une attaque sans message Pour forger une signature d'un message quelconque $m \in (\mathbb{Z}/p\mathbb{Z})^*$, un attaquant

1. choisit au hasard $(t_1, t_2) \in [\![1, q-1]\!]^2$,
2. calcule $U = \alpha^{t_1} y_A^{t_2} \mod p$,
3. puis $r = mUy_B^{t_1} \mod p$,
4. et finalement $s = -r - t_2 \mod q$.

La signature ainsi obtenue (sans la clé secrète d'Alice) (r, s, U) est valide puisque

$$r\left(y_A^{(r+s)}U\right)^{-x_B}U^{-1} = mUy_B^{t_1}\left(y_A^{-t_2}U\right)^{-x_B}U^{-1} = m\alpha^{t_1x_B}\alpha^{x_At_2x_B}\alpha^{-x_B(t_1+x_At_2)} = m.$$

Les signatures ainsi forgées sont inconditionnellement indistinguables des signatures produites par Alice.

Sekhar a déduit la sécurité de ses signatures de celle de schémas existants, sans la reprendre au début. La modification d'un schéma cache d'éventuelles failles qui ne se remarquent qu'en faisant une preuve complète. Il n'en reste pas moins que la construction de schéma de signatures à vérificateur désigné, ou à confirmeur désigné, avec recouvrement de message reste un point intéressant à étudier.

4.2.2 DVSBMH

Dans l'article présenté à SCN'04 [145], nous proposons avec Damien Vergnaud une variante du schéma DVSBM de Steinfeld *et al.* issu de [231]. L'intérêt de notre construction réside dans le fait qu'une simple remarque permet de réhabiliter les applications bilinéaires "bas-coût" comme support au schéma : en effet, nous allons montrer qu'on peut instancier le nouveau schéma DVSBMH avec l'exponentiation discrète comme application bilinéaire, ce qui est rarement possible dans les protocoles basés sur les couplages. De plus, ayant proposé une formalisation d'une notion forte d'anonymat du signataire, (voir définition 57), nous apportons à notre variante cette propriété. L'utilisation judicieuse d'une fonction de hachage rend possible l'usage de l'exponentiation discrète, et permet aussi de réduire considérablement la taille des signatures. Par ailleurs, à sécurité égale, notre schéma de signature à vérificateur désigné est le plus efficace des schémas connus, grâce à des réductions relativement fines des attaques EF-CMA et PSI-CMA aux problèmes sous-jacents.

4.2.2.1 Description

La figure 4.2 décrit le schéma DVSBMH. Cette construction nécessite deux fonctions de hachage, dont l'une envoie le message sur une courbe : on peut trouver des techniques le permettant dans [10, 38].

4.2.2.2 Avantages de cette construction

Dans le schéma DVSBM de Steinfeld *et al.*, la vérification de la signature désignée se fait en testant l'égalité de deux couplages. Ces derniers peuvent être en fait calculés indépendamment (et même en avance du côté du vérificateur, si celui-ci connaît le message à signer). En effet, si l'on note $P_A = aP$ et $P_B = bP$ les clés d'Alice et Bob, la vérification de la signature est

$$e(aH(m), P_B) = e(bH(m), P_A).$$

Bob, le vérificateur désigné, doit donc calculer l'élément $e(bH(m), P_A)$, qui est exacement une DVS dont le vérificateur serait Alice. Le schéma est en fait complètement symétrique. Il ne lui reste plus alors qu'à tester une égalité (en fait à comparer sa signature et celle d'Alice), et Alice, elle, peut donc n'envoyer qu'un haché de sa signature (en notant $g : \mathbb{H} \longrightarrow \{0,1\}^{k_2}$ la fonction de hachage) : $g(e(aH(m), P_B))$, puisque Bob peut calculer la même valeur de son côté. La vérification devient

$$g(e(aH(m), P_B)) = g(e(bH(m), P_A)).$$

Génération des paramètres publics : DVSBMH.Setup

• k_1 et k_2 sont deux paramètres de sécurité,
• $(q, P_0, \mathbb{G}_0, P_1, \mathbb{G}_1, \mathbb{H}, e)$ est généré par $\mathsf{BDHGen}(k_1)$
• Soit $[\{0,1\}^* \times \{0,1\}^{n_r} \longrightarrow \mathbb{G}_0]$ une famille de fonctions de hachage et h un élément pris au hasard dans cette famille.
• Soit $[\mathbb{H} \longrightarrow \{0,1\}^{k_2}]$ une famille de fonctions de hachage et g un élément aléatoire de cette famille.

Génération des clés du signataire : DVSBMH.SKeyGen

• $a \in [\![1, q-1]\!]$ est la clé secrète, $P_A = aP_0$ est la clé publique

Génération des clés du vérificateur : DVSBMH.VKeyGen

• $b \in [\![1, q-1]\!]$ est la clé secrète, $P_B = bP_0$ est la clé publique

Signature à vérificateur désigné : DVSBMH.Sign

• Étant donné un message m, la clé secrète de signature a, la clé publique P_B du vérificateur désigné,
• Alice calcule $H = h(m||r)$ et $s = g(e(P_B, aH))$
• la signature est $\sigma = (s, r)$.

Verification désignée : DVSBMH.Verify

• Étant donné une paire $(m, (s, r))$, la clé publique de signature P_A, et la clé secrète du vérificateur b, l'algorithme accepte la signature si et seulement si

$$s = g(e(P_A, bh(m||r))).$$

Fig. 4.2 – DVSBMH : description

Une telle utilisation de la fontion de hachage g, si elle permet de réduire significativement la taille des signatures (puisque le couplage envoie dans un sous-groupe du groupe multiplicatif d'un corps fini), rend aussi possible l'instanciation du schéma par l'exponentiation discrète, comme nous allons le montrer dans la section suivante. Le calcul d'une exponentiation discrète est beaucoup plus rapide que celui d'un couplage de type Weil ou Tate. Or il est bien clair que remplacer un couplage de type Weil par une exponentiation discrète dans DVSBM rend le schéma forgeable : en effet, si l'on possède une signature d'un message m, elle s'écrit alors $\sigma = \alpha^{abh(m)}$, et pour obtenir une signature de tout autre message m', il suffit de calculer $\sigma' = \sigma^{h(m')/h(m)}$. Pour prouver la sécurité de DVSBM, les auteurs exigent que l'isomorphisme entre $\mathbb{G}_0$ et $\mathbb{G}_1$ soit connu et calculable, ce qui n'est pas le cas de l'exponentiation discrète.

Ainsi, en termes de performance, la génération de la signature est une exponentiation, la vérification consiste en l'évaluation d'un couplage (schéma avec une meilleure sécurité), ou bien d'une exponentiation (schéma le plus efficace en pratique).

Notre schéma possède en outre la propriété d'anonymat du signataire de la définition 57.

4.2.2.3 Preuve de sécurité dans le cas général

Nous prouvons ici la sécurité de DVSBMH dans le cas où l'on utilise un générateur de groupes BDH quelconque : nous n'utilisons donc pas la connaissance explicite d'un isomorphisme ψ entre $\mathbb{G}_0$ et $\mathbb{G}_1$.

Correctness et *source hiding* La consistance de DVSBMH (que l'on choisisse un générateur de groupes BDH symétrique on non) provient immédiatement de la bilinéarité de e. En effet,

$$e(P_B, ah(m||r)) = e(bP, ah(m||r)) = e(aP, bh(m||r)) = e(P_A, bh(m||r)).$$

La propriété de source hiding est bien assurée de façon inconditionnelle.

Inforgeabilité Nous montrons que l'inforgeabilité de DVSBMH résiste aux attaques à messages choisis dans le modèle de l'oracle aléatoire [23] sous l'hypothèse que le problème GBDH est difficile. Nous utilisons essentiellement la technique de Goh et Jarecki [105] pour obtenir une réduction quadratique au problème GBDH. C'est le fait de rendre de cette façon les signatures probabilistes qui donnent (gratuitement) la propriété d'anonymat du signataire. L'idée importante, dans cette preuve, est d'utiliser l'oracle décisionnel à notre disposition pour gérer la G-List[18].

Théorème 23 (Inforgeabilité de DVSBMH) *Soit k un entier, BDHGen un générateur de groupes BDH, f_1, f_2, $f_3 : \mathbb{N} \longrightarrow \mathbb{N}$ trois fonctions, et soit DVSBMH le schéma de DVS associé. Dans le modèle de l'oracle aléatoire, pour tout adversaire EF-CMA contre DVSBMH, avec paramètre de sécurité k, probabilité de succès $\mathbf{Succ}^{\mathsf{ef-cma}}_{\mathsf{DVSBMH},\mathcal{A}} = \varepsilon$, en temps τ, et pouvant faire $q_{\mathcal{H}}$ et $q_{\mathcal{G}}$ requêtes aux oracles aléatoires, q_Σ requêtes à l'oracle de signature, et q_Υ requêtes à l'oracle de vérification, il existe un adversaire $\mathcal{D}$ contre GBDH, d'avantage $\varepsilon' = \mathbf{Succ}^{\mathsf{gbdh}}_{\mathsf{BDHGen},D}(k)$ en temps $\tau' \in \mathbb{N}$ tels que*

$$\begin{cases} \varepsilon' \geq \varepsilon - \dfrac{q_\Sigma(q_{\mathcal{H}} + q_\Sigma + 1)}{2^{n_r}} - \dfrac{(q_{\mathcal{G}} + q_\Sigma + 1)(q_\Sigma + q_\Upsilon + 1)}{2^{k_1}} - \dfrac{(q_\Upsilon + 1)(q_\Sigma + 1)}{2^{k_2}} \\ \tau' \leq \tau + (q_{\mathcal{H}} + 2q_\Sigma + 1)\left(T_{Exp-\mathbb{G}_1} + O(1)\right) + (q_{\mathcal{G}} + q_\Sigma + 1)(q_{\mathcal{G}} + q_\Sigma + q_\Upsilon)\left(T_{DDH} + O(1)\right), \end{cases}$$

où $k_1 = f_1(k)$, $k_2 = f_2(k)$ et $n_r = f_r(k)$.

Preuve : Soit k un paramètre de sécurité, et $(q, P_0, \mathbb{G}_0, P_1, \mathbb{G}_1, \mathbb{H}, e)$ généré par $\mathsf{BDHGen}(k_1)$. Soit (R_1, R_2, R_3) une instance aléatoire du problème GBDH. Nous allons décrire un algorithme $\mathcal{D}$ qui utilise comme sous-programme l'attaquant $\mathcal{A}$ contre DVSBMH pour calculer une solution de l'instance donnée.

Jeu_0 On considère un attaquant EF-CMA $\mathcal{A}$ de probabilité de succès $\mathbf{Succ}^{\mathsf{ef-cma}}_{\mathsf{DVSBMH},\mathcal{A}}$ en temps τ. Ce jeu est le jeu réel auquel participe $\mathcal{A}$. Les algorithmes de génération de clés (pour le signataire et pour le vérificateur) sont lancés une fois pour produire les couples de clés (sk_A, pk_A) et (sk_B, pk_B). $\mathcal{A}$ reçoit les clés publiques pk_A et pk_B, puis après avoir questionné $q_{\mathcal{H}}$ et $q_{\mathcal{G}}$ fois les oracles aléatoires $\mathcal{H}$ et $\mathcal{G}$, q_Σ fois l'oracle de signature Σ, et q_Υ fois l'oracle

[18] des idées similaires ont été utilisées par Dent dans [72]

de vérification Υ, il produit un couple message/signature $(m^\star, \sigma^\star)$ avec $\sigma^\star = (s^\star, r^\star)$. La signature $\sigma^\star$ n'a pas été obtenue auprès de l'oracle de signature, mais il peut vérifier celle-ci en la proposant à l'oracle de vérification.

Dans toute la suite de la démonstration, pour tout jeu Jeu_j, on note Forge_j l'événement $\mathsf{DVSBMH.Verify}(k, \text{params}, sk_B, pk_A, m^\star, \sigma^\star) = 1$.

Par définition,

$$\Pr[\mathsf{Forge}_0] = \mathbf{Succ}^{\mathsf{ef-cma}}_{\mathsf{DVSBMH},\mathcal{A}}(k).$$

Jeu_1 À partir de maintenant, l'algorithme $\mathcal{D}$ modifie certaines entrées de $\mathcal{A}$. Il commence par remplacer pk_A par R_1 et pk_B par R_2. La distribution de (pk_A, pk_B) reste inchangée, puisque (R_1, R_2, R_3) est une instance aléatoire du problème GBDH. Ainsi

$$\Pr[\mathsf{Forge}_1] = \Pr[\mathsf{Forge}_0].$$

Jeu_2 Dans ce jeu, $\mathcal{D}$ va simuler l'oracle aléatoire $\mathcal{H}$: pour toute nouvelle requête de la forme $(m, r) \in \{0,1\}^* \times \{0,1\}^{n_r}$ à $\mathcal{H}$, il tire au hasard un élément $u \in [\![1, q-1]\!]$ et calcule le point $Q = uR_3$. Il met à jour une $\mathsf{H\text{-}List}$ en y rajoutant (m, r, u, Q), et répond à la requête de $\mathcal{A}$ ou d'un oracle par Q. Dans le modèle de l'oracle aléatoire, ce jeu est identique au jeu précédent :

$$\Pr[\mathsf{Forge}_2] = \Pr[\mathsf{Forge}_1].$$

Jeu_3 $\mathcal{D}$ prend maintenant le contrôle de l'oracle aléatoire $\mathcal{G}$ et maintient une $\mathsf{G\text{-}List}$ de la façon suivante pour le simuler. Pour toute requête $\tilde{s} \in \mathbb{H}$,

- si la $\mathsf{G\text{-}List}$ contient un triplet $(\tilde{s}, \perp, s)$, $\mathcal{D}$ renvoie s comme réponse à l'oracle,
- sinon, il parcourt la $\mathsf{G\text{-}List}$ et vérifie, pour tout triplet $(\perp, u, s)$, si $(pk_A, pk_B, uP_1, \tilde{s})$ est un quadruplet Diffie-Hellman bilinéaire valide (grâce à l'oracle DBDH à sa disposition). Si c'est le cas pour un triplet, il renvoie s comme réponse.
- Dans les autres cas, il tire au hasard $s \in \{0,1\}^{k_2}$, rajoute le triplet $(\tilde{s}, \perp, s)$ à la $\mathsf{G\text{-}List}$, et renvoie s comme réponse. Ce jeu simule parfaitement l'oracle $\mathcal{G}$ et donc

$$\Pr[\mathsf{Forge}_3] = \Pr[\mathsf{Forge}_2].$$

Jeu_4 $\mathcal{D}$ va maintenant simuler l'oracle de signature : pour tout message $m \in \{0,1\}^*$ à signer, il tire au hasard trois éléments $r \in \{0,1\}^*$, $s \in \{0,1\}^{k_2}$ et $u \in [\![1, q-1]\!]$, puis calcule $Q = uP_1$.

- Si la $\mathsf{H\text{-}List}$ contient un quadruplet $(m, r, ?, ?)$, il arrête la simulation. Sinon, il lui rajoute (m, r, u, Q),
- ensuite, $\mathcal{D}$ parcourt la $\mathsf{G\text{-}List}$ et vérifie, pour tous les triplets $(\tilde{s}, \perp, ?)$, si $(pk_A, pk_B, uP_1, \tilde{s})$ est un quadruplet Diffie-Hellman bilinéaire valide, et pour tous les triplets $(\perp, v, ?)$ si $u = v$. Dans ces deux cas, il arrête la simulation,
- sinon, il rajoute $(\perp, u, s)$ dans la $\mathsf{G\text{-}List}$ et répond (r, s) à la requête de l'attaquant.

Puisqu'il y a au plus $q_{\mathcal{H}} + q_\Sigma + 1$ requêtes à l'oracle aléatoire $\mathcal{H}$ et $q_{\mathcal{G}} + q_\Sigma + 1$ à l'oracle aléatoire $\mathcal{G}$, la simulation s'arrête avec une probabilité au plus $(q_{\mathcal{H}} + q_\Sigma + 1)2^{-n_r} + (q_{\mathcal{G}} + q_\Sigma + 1)2^{-k_1}$. Sinon, l'algorithme $\mathcal{D}$ simule parfaitement les signatures. En faisant la somme pour chacune des q_Σ requêtes, on obtient

$$|\Pr[\mathsf{Forge}_4] - \Pr[\mathsf{Forge}_3]| \leq \left(\frac{(q_{\mathcal{H}} + q_\Sigma + 1)}{2^{n_r}} + \frac{(q_{\mathcal{G}} + q_\Sigma + 1)}{2^{k_1}}\right) q_\Sigma$$

Jeu_5 Dans ce jeu $\mathcal{D}$ impose à l'oracle de vérification de rejeter tous les couples message/signature $(m,(s,r))$ pour lesquels s n'a pas été obtenu auprès de $\mathcal{G}$. Dans ce cas,

$$|\Pr[\mathsf{Forge}_5] - \Pr[\mathsf{Forge}_4]| \leq \frac{q_\Upsilon + 1}{2^{k_2}}.$$

Jeu_6 Finalement, il ne reste plus à $\mathcal{D}$ que la simulation de l'oracle de vérification. Pour tous les couples message/signature $(m,(s,r))$ dont on demande une vérification, $\mathcal{D}$ commence par chercher dans la $\mathsf{H\text{-}List}$ un quadruplet $(m,r,?,?)$. S'il n'y en a pas, il rejette la signature. Sinon, la $\mathsf{H\text{-}List}$ contient un quadruplet (m,r,u,Q). Ensuite, il parcourt la $\mathsf{G\text{-}List}$: si elle contient un triplet $(\tilde{s},\perp,s)$, il accepte la signature si $(pk_A,pk_B,Q,\tilde{s})$ est un quadruplet Diffie-Hellman bilinéaire valide, sinon, la $\mathsf{G\text{-}List}$ contient un triplet $(\perp,v,s)$ et la signature est acceptée si $u=v$. Ainsi,

$$|\Pr[\mathsf{Forge}_6] - \Pr[\mathsf{Forge}_5]| \leq \left(\frac{(q_\mathcal{G} + q_\Sigma + 1)}{2^{k_1}} + \frac{q_\Sigma}{2^{k_2}}\right)(q_\Upsilon + 1).$$

À la fin du Jeu_6, l'attaquant $\mathcal{A}$ sort un couple message/signature $(m^\star,(s^\star,r^\star))$. Par définition de la forge existentielle, $\mathcal{D}$ trouve dans la $\mathsf{H\text{-}List}$ un quadruplet $(m^\star,r^\star,u^\star,Q^\star)$. D'après la simulation, $(pk_A,pk_B,Q^\star,\tilde{s}^\star)$ est un quadruplet Diffie-Hellman bilinéaire valide, et donc

$$z^\star = \tilde{s}^{\star\ (u^\star)^{-1}}$$

donne une solution à l'instance (R_1,R_2,R_3) du problème GBDH.

Ainsi, le succès de l'algorithme $\mathcal{D}$ pour résoudre le problème GBDH est

$$\mathbf{Succ}^{\mathsf{gbdh}}_{\mathsf{BDHGen},\mathcal{D}}(k) \geq \mathbf{Succ}^{\mathsf{ef-cma}}_{\mathsf{DVSBMH},\mathcal{A}}(k) - \frac{q_\Sigma(q_\mathcal{H} + q_\Sigma + 1)}{2^{n_r}} - \frac{(q_\mathcal{G} + q_\Sigma + 1)}{2^{k_1}}(q_\Sigma + q_\Upsilon + 1)$$
$$-\frac{(q_\Upsilon + 1)(q_\Sigma + 1)}{2^{k_2}},$$

et ce en temps

$$\tau' \leq \tau + (q_\mathcal{H} + 2q_\Sigma + 1)\left(T_{\mathrm{Exp}-\mathbb{G}_1} + O(1)\right) + (q_\mathcal{G} + q_\Sigma + 1)(q_\mathcal{G} + q_\Sigma + q_\Upsilon)\left(T_{DDH} + O(1)\right).$$

□

***Remarque* 10** *En insérant le haché $H = h(m||r)$ dans les entrées de la fonction de hachage g, et donc en calculant $s = g(e(P_B, aH)||H)$, on peut obtenir une réduction plus fine au problème GBDH.*

Anonymat Nous énonçons maintenant le fait que s'il existe un attaquant PSI-CMA qui lève l'anonymat du signataire dans DVSBMH, alors celui-ci peut être utilisé pour résoudre le problème GBDH.

Théorème 24 (Anonymat de DVSBMH) *Soit k un entier, BDHGen un générateur de groupes BDH, f_1, f_2, $f_3 : \mathbb{N} \longrightarrow \mathbb{N}$ trois fonctions, et soit DVSBMH le schéma de DVS associé. Dans le modèle de l'oracle aléatoire, pour tout adversaire PSI-CMA $\mathcal{A}$ contre DVSBMH, de paramètre de sécurité k, ayant un avantage $\mathbf{Adv}^{\mathsf{psi-cma}}_{\mathsf{DVSBMH},\mathcal{A}}(k) = \varepsilon$ en temps τ, et pouvant faire $q_\mathcal{H}$ et $q_\mathcal{G}$ requêtes aux oracles aléatoires, q_Σ requêtes aux oracles de signature, et q_Υ requêtes à l'oracle de* *vérification, il existe un adversaire $\mathcal{D}$ contre GBDH, d'avantage $\mathbf{Succ}^{\mathsf{gbdh}}_{BDHGen,D}(k) = \varepsilon'$ en temps τ' tel que*

$$\begin{cases} \varepsilon' \geq \dfrac{\varepsilon}{2} - \dfrac{q_\Sigma(q_\mathcal{H} + q_\Sigma + 1)}{2^{n_r}} - \dfrac{(q_\mathcal{G} + q_\Sigma + 1)(q_\Sigma + q_\Upsilon + 1)}{2^{k_1}} - \dfrac{(q_\Upsilon + 1)(q_\Sigma + 1)}{2^{k_2}} \\ \tau' \leq \tau + (q_\mathcal{H} + 2q_\Sigma + 1)\left(T_{Exp-\mathbb{G}_1} + O(1)\right) + (q_\mathcal{G} + q_\Sigma + 1)(q_\mathcal{G} + q_\Sigma + q_\Upsilon)\left(T_{DDH} + O(1)\right), \end{cases}$$

où $k_1 = f_1(k)$, $k_2 = f_2(k)$ et $n_r = f_r(k)$.

Preuve : Cette preuve s'effectue de la même manière que la précédente : l'oracle décisionnel DBDH est utilisé pour gérer la H-List. □

4.2.2.4 Preuve de sécurité dans le cas où $\mathbb{G}_0 = \mathbb{G}_1$

Nous étudions dans cette section la sécurité de la version de DVSBMH qui utilise un générateur de groupes BDH symétrique. Typiquement, on peut utiliser pour le mettre en œuvre le couplage de Weil ou Tate. L'utilisation de ces applications permet une réduction fine de la forge au problème GBDH. Il n'y a donc quasiment pas de perte de sécurité, et le schéma offre la meilleure sécurité exacte par rapport aux autres schémas existants. La perte de sécurité dans la réduction de l'anonymat à CBDH peut disparaître grâce à un oracle décisionnel. Dans le cas général étudié précédemment, la fonction de hachage g était fondamentale pour la sécurité. Ici, elle n'est pas nécessaire pour obtenir un schéma sûr, mais permet de réhausser le niveau de sécurité : l'anonymat, sans le hachage final (mais avec le “sel aléatoire” évidemment), est relié au problème DBDH, qui est plus facile que CBDH.

Inforgeabilité Le théorème suivant relie la forge existentielle du DVSBMH symétrique au problème GBDH.

Théorème 25 (Inforgeabilité de DVSBMH) *Soit k un entier, BDHGen un générateur de groupes BDH symétrique, f_1, f_2, $f_3 : \mathbb{N} \longrightarrow \mathbb{N}$ trois fonctions, et soit DVSBMH le schéma de DVS associé. Dans le modèle de l'oracle aléatoire, pour tout adversaire EF-CMA contre DVSBMH, avec paramètre de sécurité k, probabilité de succès $\mathbf{Succ}^{\mathsf{ef-cma}}_{DVSBMH,\mathcal{A}} = \varepsilon$, en temps τ, et pouvant faire $q_\mathcal{H}$ et $q_\mathcal{G}$ requêtes aux oracles aléatoires, q_Σ requêtes à l'oracle de signature, et q_Υ requêtes à l'oracle de vérification, il existe un adversaire $\mathcal{D}$ contre GBDH, d'avantage $\varepsilon' = \mathbf{Succ}^{\mathsf{gbdh}}_{BDHGen,D}(k)$ en temps $\tau' \in \mathbb{N}$ tels que*

$$\begin{cases} \varepsilon' \geq \varepsilon - \dfrac{(q_\mathcal{H} + q_\Sigma)q_\Sigma}{2^{n_r}} - (1 + q_\Upsilon)\left(\dfrac{q_\mathcal{G}}{2^{k_1}} + \dfrac{1}{2^{k_2}}\right) \\ \tau' \leq \tau + (q_\mathcal{H} + q_\Sigma)\left(T_{Exp-\mathbb{G}} + O(1)\right) + q_\Sigma\left(T_{Exp-\mathbb{H}} + O(1)\right) + (q_\Upsilon + 1)\left(T_{DBDH} + O(1)\right) \end{cases}$$

où $k_1 = f_1(k)$, $k_2 = f_2(k)$ et $n_r = f_r(k)$.

Preuve : Cette preuve est une modification de la preuve du schéma de Steinfeld *et al.* dans [231], en utilisant la technique de Goh et Jarecki [105]. □

Anonymat L'anonymat de DVSBMH symétrique repose sur la difficulté du problème CBDH. La réduction est donné dans le théorème suivant :

Théorème 26 (Anonymat de DVSBMH) *Soit k un entier, BDHGen un générateur de groupes BDH symétrique, f_1, f_2, $f_3 : \mathbb{N} \longrightarrow \mathbb{N}$ trois fonctions, et soit DVSBMH le schéma de DVS associé. Dans le modèle de l'oracle aléatoire, pour tout adversaire PSI-CMA $\mathcal{A}$ contre DVSBMH, de paramètre de sécurité k, ayant un avantage $\mathbf{Adv}^{\mathsf{psi-cma}}_{\mathsf{DVSBMH},\mathcal{A}}(k) = \varepsilon$ en temps τ, et pouvant faire $q_{\mathcal{H}}$ et $q_{\mathcal{G}}$ requêtes aux oracles aléatoires, q_{Σ} requêtes aux oracles de signature, et q_{Υ} requêtes à l'oracle de vérification, il existe un adversaire $\mathcal{D}$ contre CBDH, de probabilité de succès $\mathbf{Succ}^{\mathsf{cbdh}}_{\mathsf{BDHGen},D}(k) = \varepsilon'$ en temps τ' tel que*

$$\begin{cases} \varepsilon' \geq \dfrac{\varepsilon}{2q_{\mathcal{G}}} - \dfrac{(q_{\mathcal{H}} + q_{\Sigma} + 1)(q_{\Sigma} + 1)}{2^{n_r} q_{\mathcal{G}}} - \dfrac{q_{\Upsilon}}{2^{k_2} q_{\mathcal{G}}} - \dfrac{q_{\Upsilon}}{2^{k_1}} \\ \tau' \leq \tau + (q_{\mathcal{H}} + q_{\Sigma})(T_{Exp-\mathbb{G}} + O(1)) + (q_{\Sigma} + q_{\Upsilon})(T_{Exp-\mathbb{H}} + O(1)) \end{cases}$$

où $k_1 = f_1(k)$, $k_2 = f_2(k)$ et $n_r = f_r(k)$.

Preuve : Soit k un paramètre de sécurité, et $(q, P_0, \mathbb{G}_0, P_1, \mathbb{G}_1, \mathbb{H}, e)$ généré par $\mathsf{BDHGen}(k_1)$, avec donc $\mathbb{G}_0 = \mathbb{G}_1 (= \mathbb{G})$ et $P_0 = P_1 (= P)$. Soit (R_1, R_2, R_3) une instance aléatoire du problème CBDH. Nous allons décrire un algorithme $\mathcal{D}$ qui résout cette instance en utilisant la capacité de l'attaquant PSI-CMA $\mathcal{A}$ à distinguer 2 signataires potentiels dans DVSBMH.

Jeu_0 On considère un attaquant PSI-CMA $\mathcal{A} = (\mathsf{find}, \mathsf{guess})$ d'avantage $\mathbf{Adv}^{\mathsf{psi-cma}}_{\mathsf{DVSBMH},\mathcal{A}}(k)$ en temps τ. Dans ce jeu d'attaque réelle de $\mathcal{A}$, l'algorithme de génération de clés pour le signataire est lancé deux fois pour produire les 2 couples (sk_{A_0}, pk_{A_0}) et (sk_{A_1}, pk_{A_1}), et celui pour le vérificateur fournit la paire (sk_B, pk_B). L'adversaire $\mathcal{A}$ reçoit les clés publiques pk_{A_0}, pk_{A_1} et pk_B, et a accès, dans ses deux phases, aux oracles aléatoires $\mathcal{H}$ et $\mathcal{G}$, aux oracles de signatures Σ_0 et Σ_1, et à l'oracle de vérification Υ. À la fin de son étape find il propose un message $m^\star$, et une signature challenge est produite en tirant un bit b au hasard, et en calculant

$$(s^\star, r^\star) = \mathsf{DVSBMH.Sign}(k, \mathrm{params}, sk_{A_b}, pk_B, m^\star).$$

Étant donné cette signature et après de nouvelles discussions avec les différents oracles durant sa deuxième phase guess, $\mathcal{A}$ sort un bit b'. On note $q_{\mathcal{H}}$, $q_{\mathcal{G}}$, q_{Σ} et q_{Υ} le nombre de requêtes aux oracles aléatoires $\mathcal{H}$ et $\mathcal{G}$, aux 2 oracles de signatures et à l'oracle de vérification. L'attaquant ne peut bien sûr pas demander à vérifier la signature challenge $(r^\star, s^\star)$.

Dans toute la suite de la démonstration, pour tout jeu Jeu_j, on note Guess_j l'événement $b = b'$.

Par définition,

$$|2\Pr[\mathsf{Guess}_0] - 1| = \mathbf{Adv}^{\mathsf{psi-cma}}_{\mathsf{DVSBMH},\mathcal{A}}(k)$$

Jeu_1 L'algorithme $\mathcal{D}$ commence maintenant à modifier l'environnement de $\mathcal{A}$. Il tire un entier $\ell \in [\![1, q-1]\!]$ au hasard, et remplace les clés pk_{A_0} par R_1, pk_{A_1} par ℓR_1 et pk_B par R_2. La distribution de $(pk_{A_0}, pk_{A_1}, pk_B)$ est inchangée puisque (R_1, R_2, R_3) est pris au hasard dans $\mathbb{G}^3$ et ℓ est tiré au hasard. Ainsi

$$\Pr[\mathsf{Guess}_1] = \Pr[\mathsf{Guess}_0].$$

Jeu_2 Dans ce jeu, $\mathcal{D}$ simule les oracles aléatoires $\mathcal{H}$ et $\mathcal{G}$.

- Pour toute nouvelle requête $\tilde{s} \in \mathbb{H}$ à l'oracle $\mathcal{G}$, $\mathcal{D}$ renvoie une réponse au hasard dans $\{0,1\}^{k_2}$, et tient à jour une G-List.

- Pour toute nouvelle requête $(m, r) \in \{0,1\}^* \times \{0,1\}^{n_r}$ à l'oracle $\mathcal{H}$, il tire au hasard $u_{m,r} \in [\![1, q-1]\!]$ et calcule le point $Q_{m,r} = u_{m,r}P$. Il stocke $(m, r, u_{m,r}, Q_{m,r})$ dans la H-List, et renvoie $Q_{m,r}$ comme réponse à la requête. Dans le modèle de l'oracle aléatoire, ce jeu est identique au précédent :

$$\Pr[\mathsf{Guess}_2] = \Pr[\mathsf{Guess}_1].$$

Jeu_3 $\mathcal{D}$ modifie la génération du challenge de la façon suivante : il tire tout d'abord $r^\star \in \{0,1\}^{n_r}$ au hasard. Si la H-List contient un quadruplet $(m^\star, r^\star, u_{m^\star,r^\star}, Q_{m^\star,r^\star})$, il arrête la simulation. Sinon, il stocke $(m^\star, r^\star, \perp, R_3)$ dans la H-List, et calcule $s^\star = \mathcal{G}(e(sk_A R_3, pk_B))$. Puisqu'il y a au plus $q_{\mathcal{H}} + q_\Sigma + 1$ couples message/aléa demandés à l'oracle $\mathcal{H}$, la simulation s'est arrêtée avec une probabilité au plus $(q_{\mathcal{H}} + q_\Sigma + 1)2^{-n_r}$. Sinon, la signature est parfaitement réussie, aussi

$$|\Pr[\mathsf{Guess}_3] - \Pr[\mathsf{Guess}_2]| \leq \frac{q_{\mathcal{H}} + q_\Sigma + 1}{2^{n_r}}.$$

Jeu_4 $\mathcal{D}$ simule ici les deux oracles de signatures Σ_i $(i \in \{0,1\})$. Pour tout message $m \in \{0,1\}^*$ soumis à Σ_i, il tire au hasard $r \in \{0,1\}^{n_r}$. Si $(m, r) = (m^\star, r^\star)$, il arrête la simulation. Sinon, il invoque l'oracle $\mathcal{H}$ sur (m, r) et obtient un quadruplet $(m, r, u_{m,r}, Q_{m,r})$, puis $\mathcal{D}$ calcule $s = \mathcal{G}(e(pk_{A_i}, pk_B)^{u_{m,r}})$. Le couple (s, r) fournit une signature valide du message m. Puisqu'il y a au plus $q_{\mathcal{H}} + q_\Sigma + 1$ couples message/aléa proposés à $\mathcal{H}$, cette simulation échoue avec une probabilité au plus $(q_{\mathcal{H}} + q_\Sigma + 1) \cdot 2^{-n_r}$. En sommant sur les q_Σ requêtes, on obtient alors :

$$|\Pr[\mathsf{Guess}_4] - \Pr[\mathsf{Guess}_3]| \leq \frac{(q_{\mathcal{H}} + q_\Sigma + 1)q_\Sigma}{2^{n_r}}.$$

Jeu_5 Dans ce jeu, $\mathcal{D}$ impose que l'oracle de vérification rejette tous les triplets du type message/signature/entité $(m, (s, r), A_i)$ $(i \in \{0,1\})$ pour lesquels s n'a pas été obtenu auprès de $\mathcal{G}$. Cela ne fait une différence que si (s, r) est une signature valide sur m, alors que $\tilde{s} = e(pk_B, sk_A\mathcal{H}(m, r))$ n'a pas été proposé à $\mathcal{G}$. Puisque $\mathcal{G}(\tilde{s})$ est uniformément distribué, l'égalité $s = \mathcal{G}(\tilde{s})$ se produit avec une probabilité de 2^{-k_2}. En sommant pour chacune des q_Υ requêtes à l'oracle de vérification, on obtient

$$|\Pr[\mathsf{Guess}_5] - \Pr[\mathsf{Guess}_4]| \leq \frac{q_\Upsilon}{2^{k_2}}.$$

Jeu_6 Dans ce jeu, $\mathcal{D}$ impose que l'oracle de vérification rejette tous les triplets du type message/signature/entité $(m, (s, r), A_i)$ $(i \in \{0,1\})$ pour lesquels (m, r) n'a pas été obtenu auprès de l'oracle $\mathcal{H}$. La simulation diffère si (s, r) est une signature valide de m, alors que s a été obtenu auprès de $\mathcal{G}$, mais $\mathcal{H}(m, r)$ n'a pas été demandé. Comme $\tilde{s}$ est uniformément distribué, s a été obtenu auprès de $\mathcal{G}$ avec une probabilité au plus $q_{\mathcal{G}} \cdot 2^{-k_1}$. En sommant sur les q_Υ requêtes à l'oracle de vérification, on obtient

$$|\Pr[\mathsf{Guess}_6] - \Pr[\mathsf{Guess}_5]| \leq \frac{q_{\mathcal{G}} q_\Upsilon}{2^{k_1}}.$$

Jeu_7 $\mathcal{D}$ peut maintenant simuler l'oracle de vérification Υ. Pour tous les triplets du type message/signature/entité $(m, (s, r), A_i)$ $(i \in \{0,1\})$ pour lesquels on demande une vérification, $\mathcal{D}$ regarde si la H-List contient un quadruplet $(m, r, u_{m,r}, Q_{m,r})$ et la G-List un couple $(\tilde{s}, s)$. La signature est acceptée si $\tilde{s} = e(pk_A, pk_B)^{u_{m,r}}$). Comme la signature challenge ne peut être soumise à l'oracle de vérification, cette simulation est parfaite :

$$\Pr[\mathsf{Guess}_7] = \Pr[\mathsf{Guess}_6].$$

Jeu_8 Finalement, l'algorithme $\mathcal{D}$ tire au hasard $s^+ \in \{0,1\}^{k_2}$ et l'utilise à la place de la signature challenge $s^\star$ dans la simulation (et sans maintenir la $\mathsf{G\text{-}List}$). Cette nouvelle signature challenge n'apporte aucune information sur le bit b (au sens de la théorie de l'information), et donc

$$\Pr[\mathsf{Guess}_8] = \frac{1}{2}.$$

Ce dernier jeu est indistinguable du précédent, à moins que $e(sk_{A_b}\mathcal{H}(m^\star, r^\star), pk_B)$ soit proposé à $\mathcal{G}$ par l'oracle de signature, celui de vérification, ou par l'adversaire. Le premier cas a été écarté par dans le jeu 4, et, par définition de la sécurité $\mathsf{PSI\text{-}CMA}$, le deuxième ne peut se produire. En sortant un élément $\tilde{s}$ de la $\mathsf{G\text{-}List}$, pour calculer $\tilde{s}^{1/l^b}$, la probabilité de résoudre notre instance du problème CBDH est majorée par la probabilité que $e(sk_{A_b}\mathcal{H}(m^\star, r^\star), pk_B)$ (*i.e.* $e(P,P)^{\log_P(R_1)\log_P(R_2)\log_P(R_3)\ell^b}$) soit demandé à l'oracle $\mathcal{G}$, multipliée par $q_{\mathcal{G}}^{-1}$. Ainsi,

$$|\Pr[\mathsf{Guess}_8] - \Pr[\mathsf{Guess}_7]| \leq q_{\mathcal{G}} \mathbf{Succ}^{\mathsf{cbdh}}_{\mathcal{G}en,D}(k).$$

Finalement, on obtient

$$\mathbf{Succ}^{\mathsf{cbdh}}_{\mathsf{BDHGen},D}(k) \geq \frac{\mathbf{Adv}^{\mathsf{psi-cma}}_{\mathsf{DVSBMH},\mathcal{A}}(k)}{2q_{\mathcal{G}}} - \frac{(q_{\mathcal{H}} + q_\Sigma + 1)(q_\Sigma + 1)}{2^{n_r} q_{\mathcal{G}}} - \frac{q_\Upsilon}{2^{k_2} q_{\mathcal{G}}} - \frac{q_\Upsilon}{2^{k_1}}$$

en temps

$$\tau' \leq \tau + (q_{\mathcal{H}} + q_\Sigma)(T_{\mathrm{Exp}-\mathbb{G}} + O(1)) + (q_\Sigma + q_\Upsilon)(T_{\mathrm{Exp}-\mathbb{H}} + O(1))$$

□

***Remarque* 11** *En utilisant un oracle décisionnel dans l'étape final, on peut éliminer le facteur $q_{\mathcal{G}}^{-1}$ pour avoir une réduction fine au problème GBDH (on ne sait pas naturellement distinguer le bon élément dans la G-List).*

4.2.3 Comparaison des schémas

Nous dressons ici deux tableaux comparatifs des différents schémas de signature à vérificateur désigné. Ils rassemblent le schéma (noté SKM) de Saeednia, Kremer et Markowitch de [211], les 3 schémas de DVS universelles basés sur Schnorr et RSA de Steinfeld, Wang et Pieprzyk [232] (SchDVS_1, SchDVS_2 et RSADVS), le schéma DVSBM de Steinfeld, Bull, Wang et Pieprzyk [231], et le schéma proposé dans notre article à SCN'04 [145], instancié avec l'exponentiation discrète ou le couplage de Weil ou Tate.

Pour faire les calculs, nous faisons l'hypothèse que l'algorithme de Miller [171] [173] pour calculer un couplage de Tate sur une courbe supersingulière de degré MOV 6 sur un corps de base de taille 171 bits est 15 fois plus lent que le calcul d'une exponentiation de taille 160 bits dans un corps premier de taille 1024 bits, ce qui semble raisonnable, au vu des progrès dans les évaluations de couplages (voir par exemple [11, 12, 218]).

Sécurité fixée Dans la figure 4.3, pour chaque schéma considéré, sont précisés le problème difficile sous-jacent (pour la forge), la perte de sécurité par rapport à ce problème (en utilisant les résultats de sécurité exacte des articles correspondants), les temps de calculs pour la signature et la vérification, la taille des signatures, et les propriétés éventuelles d'anonymat du signataire, et d'universalité. Le niveau de sécurité considéré est 2^{80}. L'unité choisie est le temps de calcul de DVSBMH avec couplage. Il est à noter que d'après la remarque 10, on peut obtenir une meilleure réduction à GDH dans le cas de l'exponentiation discrète.

						DVSBMH	
Schéma	SKM	SchDVS_1	SchDVS_2	RSADVS	DVSBM	Weil/Tate	Exp. Disc.
Problème	GDH	GDH	DL	RSA	GBDH	GBDH	GDH
Perte sécu.	2^{79}	2^{99}	2^{209}	2^{149}	2^{47}	2^{8}	2^{79}
Temps calcul Sign.	3.6	14.4	114	5346	36.7	**1**	3.6
Temps calcul Verif.	7.2	7.2	342	2673	36.7	**1**	3.6
Taille sign. (en bits)	954	22942	50009	264507	4096	271	271
Anonymat	✓	✓				✓	✓
Universelles		✓	✓	✓	✓	✓	

FIG. 4.3 – Comparaison entre quelques DVS à sécurité égale (2^{80})

Sécurité industrielle L'utilisation dans la vraie vie des protocoles cryptographiques tient compte de l'existence ou non de preuves de sécurité, mais il est rare que les conclusions issues de la réduction donnée dans la preuve soient prises en compte. En effet, les pertes de sécurité issues de la réduction devraient impliquer l'utilisation de clés plus longues pour une sécurité fixée, par rapport à une réduction optimale, ce qui n'est pas respecté en pratique. La figure 4.4 compare divers schémas de signature à vérificateur désigné en oubliant les réductions exactes, et les rapporte simplement au problème sous-jacent. Cette fois-ci, c'est le temps de calcul de DVSBMH avec exponentiation discrète qui tient lieu d'unité.

						DVSBMH	
Schéma	SKM	SchDVS_1	SchDVS_2	RSADVS	DVSBM	Weil/Tate	Exp. Disc.
Temps calcul Sign.	1	2	3	23	16	16	**1**
Temps calcul Verif.	2	1	2	12	16	16	**1**
Taille sign. (en bits)	480	2048	1504	11584	1024	271	271

FIG. 4.4 – Comparaison entre quelques DVS (sécurité industrielle)

4.3 Au pluriel : signatures à vérificateurs désignés

Il paraît naturel d'étendre la possibilité de désigner un vérificateur à plusieurs entités. Y. Desmedt suggère cette généralisation à la rump session de Crypto'03 [73]. Dès leur article de 1996 [125], Jakobsson *et al.* ont esquissé une extension possible de leur schéma. Ce dernier ne protège pas l'anonymat du signataire, comme leur protocole initial. Nous avons donné la définition de signatures à vérificateurs désignés et le modèle de sécurité dans la partie 4.1.1, qui provient de l'article paru à ICICS'04 [144], où nous apportions la formalisation de ce concept. Hormis le schéma de Jakobsson *et al.*, les autres schémas DVS ne s'étendent pas naturellement à un contexte de groupe de vérificateurs (et en particulier les MAC).

Bien sûr, un signataire pourrait calculer n signatures à vérificateur désigné, puis les envoyer toutes à destination des vérificateurs, mais cette solution triviale n'est pas satisfaisante, puisqu'elle est gourmande en bande passante, en temps de calcul, tant en signature qu'en vérification.

Dans cette partie, nous proposons d'abord une construction générique d'un schéma de signatures à vérificateurs désignés grâce à des signatures d'anneau basées sur le logarithme discret, en ce sens que la clé publique du signataire est un élément du groupe sous-jacent et la clé secrète son logarithme discret. Comme nous l'avons vu dans la partie 3.2.2.2, de telles signatures d'anneau existent ([248], [1], [122],. . .). Contrairement à l'idée qu'en avait Desmedt, la taille des signatures ainsi produites ne croît pas linéairement avec le nombre de vérificateurs.

Ensuite, nous proposons un schéma de signatures pour 2 vérificateurs désignés qui est anonyme pour le signataire, en reposant sur l'échange de clés tripartite de Joux [127, 130]

Finalement, nous proposons une construction de MDVS qui possède la propriété d'anonymat du signataire sans chiffrement, pour un nombre quelconque de vérificateurs. Cette propriété est obtenue au prix d'une taille des signatures qui croît linéairement avec le nombre de vérificateurs

4.3.1 Construction générique à partir de signatures d'anneaux

Le contexte de signature d'anneau se prête bien, *a priori*, à la notion de signature à vérificateurs désignés. L'idée est d'apporter naturellement à un schéma de DVS la propriété inhérente d'ambiguïté du signataire des signatures d'anneau à deux membres. Si l'on note $P_{B_i} = b_i P$ la clé publique de l'utilisateur B_i, associée à la clé secrète b_i, l'utilisation d'un schéma basé sur le logarithme discret permet de voir la réunion des vérificateurs comme un deuxième membre dont la clé publique est la somme des clés publiques $P_B = P_{B_1} + \cdots + P_{B_n}$, et la clé secrète la somme des clés secrètes $b = b_1 + \cdots b_n$. En utilisant par exemple des techniques de calculs multi-partites, ils sont capables de produire des signatures associées à cette clé secrète.

4.3.1.1 Description

On considère dans cette construction générique un schéma de signature d'anneau

$$\mathsf{RING} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$$

qui va servir de fondation à nos signatures à vérificateurs désignés. On suppose que ce schéma de signature d'anneau est basé sur le logarithme discret, au sens indiqué précédemment. La figure 4.5 décrit le schéma générique MDVS.

4.3.1.2 À propos de MDVS

Cette construction ne protège pas l'anonymat du signataire : elle hérite naturellement des propriétés du schéma de signatures d'anneau sous-jacent, puisque l'idée est bien de voir la réunion des vérificateurs désignés comme le deuxième utilisateur de l'anneau (avec Alice). La propriété d'ambiguïté du signataire est alors immédiatement vérifiée : les n vérificateurs désignés peuvent coopérer (en utilisant par exemple du calcul multi-partite - *secure multiparty computation* - introduit par Yao en 1982 [245]) pour produire la clé publique

$$P_B = \sum_{i=1}^{n} P_{B_i} = \left(\sum_{i=1}^{n} b_i\right) P$$

"correspondant" à la clé secrète $\sum_{i=1}^{n} b_i$. C'est ce fait, ajouté à la propriété d'ambiguïté du signataire des signatures d'anneau qui assure la propriété de *source hiding* pour le schéma de signature à vérificateurs désignés : un vérificateur, sachant qu'il n'a pas participé à la génération d'une

Génération des paramètres publics : $\mathsf{MDVS.Setup}$

$\mathsf{MDVS.Setup} = \mathsf{RING.Setup}$

Génération des clés du signataire : $\mathsf{MDVS.SKeyGen}$

$\mathsf{MDVS.SKeyGen} = \mathsf{RING.KeyGen}$
On note a la clé secrète du signataire et $P_A = aP$ sa clé publique.

Génération des clés du vérificateur : $\mathsf{MDVS.VKeyGen}$

$\mathsf{MDVS.SKeyGen} = \mathsf{RING.KeyGen}$
On note b_i la clé secrète d'un vérificateur et $P_{B_i} = b_i P$ sa clé publique, pour $i \in [\![1, n]\!]$.

Signature à vérificateurs désignés : $\mathsf{MDVS.Sign}$

Étant donné un message m, la clé secrète de signature a, les clés publiques P_{B_i}, pour $i \in [\![1, n]\!]$ des n vérificateurs désignés, la signature σ de m est produite de la façon suivante :

$$\sigma = \mathsf{RING.Sign}(m, P_A, \sum_{i=1}^{n} P_{B_i}, a).$$

Vérification désignée : $\mathsf{MDVS.Verify}$

Étant donné une paire (m, σ), la clé publique de signature P_A, la clé secrète d'un vérificateur désigné b_{i_0} et les clés publiques de tous les vérificateurs, un vérificateur se convainc de la validité de la signature en s'assurant que

$$\mathsf{RING.Verify}(m, \sigma, P_A, \sum_{i=1}^{n} P_{B_i}) = 1$$

FIG. 4.5 – MDVS : description

signature, sait que c'est bien Alice qui l'a produite. Ce schéma est aussi efficace que le schéma de signature d'anneau sous-jacent.

Pour protéger l'anonymat du signataire, il faut *a priori* chiffrer la signature (*cf.* la proposition 7), il est alors nécessaire d'utiliser un échange de clé multi-partite, ce qui rend le schéma beaucoup moins efficace : dans ce cas, le schéma perd sa spontanéité.

4.3.2 Un schéma efficace dans le cas $n = 2$

En cryptographie multi-acteurs le cas de 3 utilisateurs est très souvent rencontré. Dans cette partie, nous proposons un schéma de signatures à 2 vérificateurs efficace, qui protège l'anonymat du signataire, sans qu'il y ait besoin de chiffrer la signature.

4.3.2.1 Description

La figure 4.6 décrit le schéma B2DVS de signatures à 2 vérificateurs désignés. Le schéma repose sur les signatures d'anneau de Boneh *et al.* [37], et sur l'échange de clé tripartite de Joux [127,130]. Il s'inspire des contructions génériques basées sur les signatures d'anneaux présentées dans le paragraphe précédent.

Le schéma B2DVS est particulièrement efficace en signature, puisqu'il n'y a que trois multiplications scalaires sur une courbe à calculer, et une exponentiation dans un corps fini. La taille des signatures est courte, puisqu'il s'agit de deux points sur une courbe et un aléa, ce qui en pratique, donne des signatures d'environ 480 bits. La vérification, quant à elle, nécessite le calcul de trois couplages et d'une exponentiation. Cela reste tout à fait raisonnable compte tenu des progrès qui sont réalisés dans le domaine de l'évaluation des couplages.

4.3.2.2 Preuves de sécurité

Consistance et *source hiding* La consistance de B2DVS se vérifie aisément et la propriété d'ambiguïté du signataire se déduit de celle du schéma de signature d'anneau sous-jacent (de Boneh *et al.* [37]), jointe à la construction de la clé P_{BC} comme somme des deux clés publiques P_B et P_C des vérificateurs : il leur est possible, en coopérant, de produire des signatures indistinguables de celles d'Alice en utilisant la clé secrète associée à P_{BC}, à savoir $b + c$.

Inforgeabilité L'inforgeabilité de B2DVS repose sur la difficulté du problème CDH, ce qui est énoncé dans le théorème suivant. L'attaquant EF-CMA permet en fait de résoudre directement le problème CDH' suivant, qui, au prix de 2 résolutions de ce problème, permet de résoudre CDH. Ceci explique le carré dans la probabilité du succès de l'attaquant contre CDH, et le facteur 2 pour le temps.

Problème CDH' : soit $\mathbb{G}$ un groupe engendré par $P \in \mathbb{G}$ d'ordre un grand premier q. Étant donné $(xP, xyP) \in \mathbb{G}^2$, calculer $yP \in \mathbb{G}$.

Supposons que l'on dispose d'un algorithme $\mathcal{M}$ qui résolve CDH'. Soit xP, yP une instance de CDH à résoudre. Dans un premier temps, $\mathcal{M}$ calcule, à partir de xP et $P = xx^{-1}P$, le point $x^{-1}P$. Ensuite, à partir de $x^{-1}P$ et $yP = x^{-1}xyP$, il renvoie le point xyP qui est bien la solution de l'instance initiale de CDH.

Théorème 27 (Inforgeabilité de B2DVS) *Soit k un entier, BDHGen un générateur de groupes BDH symétrique, et soit B2DVS le schéma de signatures à 2 vérificateurs désignés associé. Dans le modèle de l'oracle aléatoire, pour tout adversaire EF-CMA $\mathcal{A}$ contre B2DVS, de paramètre de sécurité k, ayant une probabilité de succès $\mathbf{Succ}^{\mathsf{ef-cma}}_{\mathsf{B2DVS},\mathcal{A}}(k) = \varepsilon$ en temps τ, et pouvant faire $q_{\mathcal{H}}$ requêtes a l'oracle aléatoire et q_Σ requêtes à l'oracle de signature, il existe un adversaire $\mathcal{D}$ contre CDH, de succès $\mathbf{Succ}^{\mathsf{cdh}}_{\mathsf{BDHGen},D}(k) = \varepsilon'$ en temps τ' tel que*

$$\begin{cases} \varepsilon' \geq \left(\dfrac{1}{2}\varepsilon - \dfrac{q_{\mathcal{H}} q_\Sigma + 1}{2^k}\right)^2 \\ \tau' \leq 2\big(\tau + (q_{\mathcal{H}} + 2q_\Sigma + O(1))T_M + q_\Sigma T_{\mathbb{H}}\big). \end{cases}$$

PREUVE :

Soit k un paramètre de sécurité, et $(q, P, \mathbb{G}, \mathbb{H}, e)$ généré par $\mathsf{BDHGen}(k)$. Soit (R_x, R_{xy}) une instance aléatoire du problème CDH', *i.e.* $R_x = xP$ et $R_{xy} = xyP$ avec x et y dans $[\![1, q]\!]$. Nous allons décrire un algorithme $\mathcal{D}$ qui calcule le point $R_y = yP$.

Génération des paramètres publics : B2DVS.Setup

- k est un paramètre de sécurité
- $(q, P, \mathbb{G}, \mathbb{H}, e)$ est généré par $\mathsf{BDHGen}(k)$
- Soit $[\{0,1\}^* \times \mathbb{H} \longrightarrow \mathbb{G}]$ une famille de fonctions de hachage h un élément pris au hasard dans cette famille.

Génération des clés du signataire : B2DVS.SKeyGen

- $a \in [\![1, q-1]\!]$ est la clé secrète, $P_A = aP$ est la clé publique

Génération des clés du vérificateur : B2DVS.VKeyGen

- $b \in [\![1, q-1]\!]$ est la clé secrète de Bob, $P_B = bP$ est sa clé publique
- $c \in [\![1, q-1]\!]$ est la clé secrète de Cindy, $P_C = cP$ est sa clé publique

Signature à vérificateurs désignés : B2DVS.Sign

- Étant donné un message m, la clé secrète de signature a, les clés publiques P_B et P_C des vérificateurs désignés,
- Alice tire au hasard $(r, \ell) \in [\![1, q-1]\!]^2$
- elle calcule $P_{BC} = P_B + P_C$, $u = e(P_B, P_C)^a$ et $M = h(m, u^\ell)$
- elle pose $Q_A = a^{-1}(M - rP_{BC})$ et $Q_{BC} = rP$.
- La signature de m est $\sigma = (Q_A, Q_{BC}, \ell)$.

Vérification désignée : B2DVS.Verify

- Étant donné une paire (m, σ), la clé publique de signature P_A, et la clé secrète de Bob b ou de Cindy c,
- Bob (resp. Cindy) calcule $u = e(P_A, P_C)^b$ (resp. $u = e(P_A, P_B)^c$), $M = h(m, u^\ell)$ et $P_{BC} = P_B + P_C$
- La signature est acceptée si

$$e(Q_A, P_A)e(Q_{BC}, P_{BC}) = e(M, P).$$

FIG. 4.6 – B2DVS : description

Jeu$_0$ On considère un attaquant EF-CMA $\mathcal{A}$ de probabilité de succès $\mathbf{Succ}^{\mathsf{ef-cma}}_{\mathsf{B2DVS},\mathcal{A}}$ en temps τ. On considère tout d'abord le jeu réel de $\mathcal{A}$. L'algorithme de génération de clés pour le signataire est lancé une fois pour produire le couple (P_A, a), et celui pour le vérificateur est lancé deux fois pour produire les couples de clés (P_B, b) et (P_C, c). $\mathcal{A}$ reçoit les clés publiques P_A, P_B et P_C, puis questionne $q_{\mathcal{H}}$ fois l'oracle aléatoire $\mathcal{H}$ et q_Σ fois l'oracle de signature Σ, et q_Υ fois l'oracle de vérification Υ. En outre, il choisit de corrompre Bob ou Cindy, puis, à l'issue de ces discussions, il produit un couple message/signature $(m^\star, \sigma^\star)$. Par définition, la signature $\sigma^\star$ n'a pas été obtenue auprès de l'oracle de signature. Ayant corrompu Bob ou Cindy, il connaît en particulier la clé commune $e(P,P)^{abc}$, et peut ainsi vérifier par lui-même toutes les signatures.

Dans toute la suite de la démonstration, pour tout jeu Jeu_j, on note Forge_j l'événement $\mathsf{B2DVS.Verify}(k, \text{params}, x, P_A, P_B, P_C, m^\star, \sigma^\star) = 1$, avec $x = b$ ou $x = c$.

Par définition,

$$\Pr[\mathsf{Forge}_0] = \mathbf{Succ}^{\mathsf{ef-cma}}_{\mathsf{DVSBMH},\mathcal{A}}(k).$$

Jeu_1 Dans ce jeu, $\mathcal{D}$ tire au hasard un indice $i_0 \in \{B, C\}$ et un entier $\alpha \in [\![1, q]\!]$. On note i_1 l'élément de $\{B, C\}$ qui n'est pas i_0 : $i_1 \in \{B, C\} \setminus \{i_0\}$. $\mathcal{D}$ tire ensuite un élément $\beta \in [\![1, q]\!]$. Finalement, il remplace la clé publique du signataire P_A par R_x, P_{i_1} par βP et P_{i_0} par $\alpha R_x - P_{i_1}$. La distribution de (P_A, P_B, P_C) reste inchangée, puisque R_x, α et β, sont tirés au hasard. Ainsi

$$\Pr[\mathsf{Forge}_1] = \Pr[\mathsf{Forge}_0].$$

Jeu_2 Dans celui-ci, $\mathcal{D}$ arrête la simulation à chaque fois que l'attaquant tente de corrompre le vérificateur i_0, sinon, il donne à l'attaquant β comme le secret du vérificateur i_1, et donc :

$$\Pr[\mathsf{Forge}_2] = \frac{1}{2}\Pr[\mathsf{Forge}_1].$$

Jeu_3 Il s'agit à présent de simuler l'oracle aléatoire $\mathcal{H}$. Pour toute nouvelle requête $(m, v) \in \{0,1\}^* \times \mathbb{G}$ à l'oracle $\mathcal{H}$, $\mathcal{D}$ tire au hasard un entier $h \in [\![1, q]\!]$, et calcule le point $M = hR_{xy}$. Le quadruplet (m, v, h, M) est stocké dans la liste de hachés $\mathsf{H\text{-}List}$ et $\mathcal{D}$ retourne M comme réponse de la part de l'oracle. Dans le modèle de l'oracle aléatoire, ce jeu est identique au jeu précédent, et ainsi on obtient

$$\Pr[\mathsf{Forge}_3] = \Pr[\mathsf{Forge}_2].$$

Jeu_4 Dans ce jeu, $\mathcal{D}$ ne garde que les exécutions qui se terminent par un couple message/signature $(m, (Q_A, Q_{BC}, \ell))$ valide pour lequel (m, u^ℓ) a été demandé à l'oracle aléatoire. Puisque $\mathcal{H}(m, u^\ell)$ est uniformément distribué, l'égalité $e(Q_A, P_A)e(Q_{BC}, P_{BC}) = e(\mathcal{H}(m, u^\ell), P)$ se produit avec une probabilité au plus 2^{-k}. Ainsi,

$$|\Pr[\mathsf{Forge}_4] - \Pr[\mathsf{Forge}_3]| \leq \frac{1}{2^k}.$$

Jeu_5 Finalement, $\mathcal{D}$ simule l'oracle de signature : pour toute requête de signature sur le message m, il tire au hasard un élément $a_2 \in [\![1, q]\!]$ et un couple $(\ell, r) \in [\![1, q]\!]^2$, et calcule $a_1 = r - a_2\alpha$. Si la $\mathsf{H\text{-}List}$ contient un quadruplet du type $(m, u^\ell, ?, ?)$, il arrête la simulation, sinon, il insère le quadruplet (m, u^ℓ, r, rR_x) dans la $\mathsf{H\text{-}List}$ et, après avoir posé $Q_A = a_1 P$ et $Q_{BC} = a_2 P$, le triplet (Q_A, Q_{BC}, ℓ) fournit une signature valide pour le message m. Si la simulation ne s'arrête pas, $\mathcal{D}$ a parfaitement simulé l'oracle de signature. Puisqu'elle s'arrête avec une probabilité au plus $q_{\mathcal{H}} 2^{-k}$, on obtient

$$|\Pr[\mathsf{Forge}_5] - \Pr[\mathsf{Forge}_4]| \leq \frac{q_{\mathcal{H}} q_\Sigma}{2^k}.$$

À l'issue du jeu numéro 5, l'attaquant $\mathcal{A}$ produit une forge $(m^\star, Q^\star_A, Q^\star_{BC}, \ell^\star)$. Par définition, il existe dans la $\mathsf{H\text{-}List}$ un quadruplet $(m^\star, v^\star, h^\star, M^\star)$. On vérifie alors que

$$\begin{array}{rcl} h^{\star-1}(Q^\star_A + \alpha Q^\star_{BC}) & = & h^{\star-1}(x^{-1}(M^\star - r^\star P_{BC}) + \alpha Q^\star_{BC}) \\ & = & h^{\star-1}(x^{-1}(h^\star R_{xy} - r^\star P_{BC}) + \alpha r^\star P) \\ & = & h^{\star-1}(x^{-1}(h^\star R_{xy} - r^\star \alpha R_x) + \alpha r^\star P) = R_y. \end{array}$$

Finalement, en tenant compte des différentes pertes, et du fait qu'il faille rejouer la simulation pour résoudre CDH, on obtient

$$\mathbf{Succ}^{\mathsf{cdh}}_{\mathsf{Game}_5}(k) \geq \left(\frac{1}{2}\mathbf{Adv}^{\mathsf{ef-cma}}_{\mathtt{B2DVS}_B,\mathcal{A}} - \frac{q_{\mathcal{H}}q_{\Sigma}+1}{2^k}\right)^2,$$

en temps <tt>$\tau' \leq 2\big(\tau + (q_{\mathcal{H}} + 2q_{\Sigma} + O(1))T_M + q_{\Sigma}T_{\mathbb{H}}\big)$.

□

Anonymat Le théorème suivant montre que l'existence d'un attaquant PSI-CMA qui réussit à lever l'anonymat du signataire dans B2DVS implique l'existence d'un attaquant qui résout le problème GBDH.

Théorème 28 (Anonymat de B2DVS) *Soit k un entier et BDHGen un générateur de groupes BDH symétrique et soit B2DVS le schéma de signatures à 2 vérificateurs désignés associé. Dans le modèle de l'oracle aléatoire, pour tout adversaire PSI-CMA $\mathcal{A}$ contre B2DVS, de paramètre de sécurité k, ayant un avantage $\mathbf{Adv}^{\mathsf{psi-cma}}_{\mathit{B2DVS},\mathcal{A}}(k) = \varepsilon$ en temps τ, et pouvant faire $q_{\mathcal{H}}$ requêtes à l'oracle aléatoire, q_{Σ} requêtes aux oracles de signature, et q_{Υ} requêtes à l'oracle de vérification, il existe un adversaire $\mathcal{D}$ contre GBDH, de succès $\mathbf{Succ}^{\mathsf{gbdh}}_{\mathit{BDHGen},D}(k) = \varepsilon'$ en temps τ' tel que*

$$\left\{\begin{array}{lll} \varepsilon' & \geq & \frac{1}{2}\varepsilon - \frac{q_{\Upsilon}}{2^k} - \frac{(q_{\mathcal{H}}+q_{\Sigma})q_{\Sigma}}{2^k} \\ \tau' & \leq & \tau + \big((q_{\mathcal{H}}+q_{\Sigma})^2 + q_{\mathcal{H}}\big)\big(T_{DBDH} + T_{\mathbb{H}} + O(1)\big) + \\ & & q_{\Upsilon}\big(3T_P + T_{\mathbb{H}} + O(1)\big) + q_{\Sigma}\big(4T_{\mathbb{G}} + O(1)\big). \end{array}\right.$$

Preuve :

Soit k un paramètre de sécurité, et $(q, P, \mathbb{G}, \mathbb{H}, e)$ généré par $\mathsf{BDHGen}(k)$. Soit (X, Y, Z) une instance aléatoire du problème GBDH, *i.e.* $X = xP$ et $Y = yP$ et $Z = zP$ avec x, y et z dans $[\![1, q]\!]$. Nous allons décrire un algorithme $\mathcal{D}$ qui calcule $e(P,P)^{xyz}$.

Jeu_0 On considère un attaquant PSI-CMA $\mathcal{A} = (\mathsf{find}, \mathsf{guess})$ d'avantage $\mathbf{Adv}^{\mathsf{psi-cma}}_{\mathsf{B2DVS},\mathcal{A}}$ en temps τ. Ce premier jeu est le vrai scénario d'attaque. Deux paires de clés (b, P_B) et (c, P_C) sont produites par l'algorithme de génération de clés pour les vérificateurs, et il en est de même du côté du signataire : deux paires de clés (a_0, P_{A_0}) et (a_1, P_{A_1}) sont générées aléatoirement par l'algorithme de génération de clés pour les signataires. Dans sa première phase find, $\mathcal{A}$ reçoit les clés publiques précédemment générées, et à l'issue de discussions avec les oracles de hachage $\mathcal{H}$, signature Σ_0 et Σ_1, et vérification Υ, il propose un message $m^\star$ au challenger. Ce dernier engendre une signature en tirant un bit $b \in \{0, 1\}$ à pile-ou-face et en appliquant l'algorithme de signature de la façon suivante : $\sigma^\star = \mathsf{B2DVS.Sign}(k, \mathrm{params}, a_b, P_{A_b}, P_B, P_C, m^\star)$. Cette signature est alors proposée à $\mathcal{A}$ qui va tenter de trouver le bit b. Il a de nouveau un accès permanent aux différents oracles. À la fin de cette deuxième étape guess, l'attaquant sort un bit $b^\star$.
On note $q_{\mathcal{H}}$ le nombre de requêtes à l'oracle aléatoire, $q_{\Sigma} = q_{\Sigma_0} + q_{\Sigma_1}$ le nombre de requêtes aux deux oracles de signatures, et q_{Υ} le nombre de requêtes à l'oracle de vérification. On note Guess_0 l'événement $b^\star = b$, et on fait de même pour l'événement Guess_i dans le Jeu_i.

Par définition,

$$|2\Pr[\mathsf{Guess}_0] - 1| = \mathbf{Adv}^{\mathsf{psi-cma}}_{\mathsf{B2DVS},\mathcal{A}}(k)$$

Jeu_1 Un entier $\alpha \in [\![1, q]\!]$ est tiré au hasard par $\mathcal{D}$, et la simulation est modifiée en remplaçant P_{A_0} par X, P_{A_1} par αX, P_B par Y et P_C par Z. La distribution des clés publiques $(P_{A_0}, P_{A_1}, P_B, P_C)$ reste inchangée, puisque on les remplace par une instance aléatoire de GBDH, et que α est choisi au hasard. Ainsi,

$$\Pr[\mathsf{Guess}_1] = \Pr[\mathsf{Guess}_0].$$

Jeu_2 Dans ce jeu, $\mathcal{D}$ simule l'oracle aléatoire $\mathcal{H}$ en maintenant une liste de hachés $\mathsf{H\text{-}List}$ de la façon suivante. Pour toute requête du type $(m, v) \in \{0,1\}^* \times \mathbb{H}$,

- si la $\mathsf{H\text{-}List}$ contient déjà un quadruplet de la forme $(m, v, \perp, M)$, $\mathcal{D}$ renvoie M comme réponse de l'oracle aléatoire,
- sinon, il parcourt la $\mathsf{H\text{-}List}$ en testant, pour les quadruplets de la forme $(m, \perp, \ell, M)$, si $(X, Y, Z, v^{1/\ell})$ est un quadruplet Diffie-Hellman bilinéaire valide. Si c'est le cas, il renvoie M comme réponse de l'oracle.
- Dans les autres cas, il tire au hasard un élément $M \in \mathbb{G}$, rajoute $(m, v, \perp, M)$ dans la $\mathsf{H\text{-}List}$, et renvoie M comme réponse.

Dans le modèle de l'oracle aléatoire, $\mathcal{D}$ a parfaitement simulé l'oracle $\mathcal{H}$, et donc :

$$\Pr[\mathsf{Guess}_2] = \Pr[\mathsf{Guess}_1].$$

Jeu_3 Dans ce jeu, $\mathcal{D}$ simule les oracles de signatures, Σ_0 et Σ_1 : pour tout message $m \in \{0,1\}^*$ proposé à l'oracle de signature Σ_i (pour $i \in \{0,1\}$) par l'attaquant ou le challenger, il tire au hasard $(q_A, q_B) \in [\![1, q-1]\!]^2$ et $\ell \in [\![1, q-1]\!]$. Il pose ensuite $M = q_A P_{A_i} + q_B P_B$, $Q_A = q_A \alpha^i P$ et $Q_B = q_B P$.

- Si la $\mathsf{H\text{-}List}$ contient un quadruplet $(m, \perp, \ell\alpha^i, ?)$, $\mathcal{D}$ arrête la simulation,
- sinon, il parcourt la $\mathsf{H\text{-}List}$ en vérifiant, pour tous les quadruplets $(m, v, \perp, ?)$ si $(X, Y, Z, v^{1/\ell})$ est un triplet Diffie-Hellman bilinéaire valide. Si c'est le cas, il arrête la simulation.
- autrement, il rajoute $(m, \perp, \ell\alpha^i, M)$ à la $\mathsf{H\text{-}List}$ et la signature simulée du message m est (Q_A, Q_B, ℓ).

Il y a au plus $q_{\mathcal{H}} + q_\Sigma$ requêtes à l'oracle aléatoire $\mathcal{H}$, cette nouvelle simulation a donc une probabilité au plus $(q_{\mathcal{H}} + q_\Sigma)2^{-k}$ de s'arrêter. Dans le cas contraire, elle simule parfaitement les oracles de signatures. En faisant la somme des probabilités, on obtient :

$$|\Pr[\mathsf{Guess}_3] - \Pr[\mathsf{Guess}_2]| \leq \frac{(q_{\mathcal{H}} + q_\Sigma)q_\Sigma}{2^k}.$$

Jeu_4 La simulation de l'oracle de vérification se fait ici de la façon suivante : pour tout triplet message/signature/entité $(m, (Q_A, Q_B, \ell), A_i)$ (avec $i \in \{0,1\}$) pour lequel l'attaquant souhaite une vérification,

- si la $\mathsf{H\text{-}List}$ ne contient pas un quadruplet de la forme $(m, ?, ?, M)$ $\mathcal{D}$ rejette la signature,
- sinon, si la $\mathsf{H\text{-}List}$ contient un quadruplet $(m, \perp, \ell, M)$, il accepte la signature si et seulement si $e(M, P) = e(Q_A, P_{A_i})e(Q_B, P_B)$,
- sinon, si la $\mathsf{H\text{-}List}$ contient un quadruplet $(m, v, \perp, M)$, il accepte la signature si et seulement si $(X, Y, Z, v^{1\ell})$ est un quadruplet Diffie-Hellman bilinéaire valide, et $e(M, P) = e(Q_A, P_{A_i})e(Q_B, P_B)$.

Une différence par rapport au jeu précédent se produit seulement si, dans la première étape, (Q_A, Q_B, ℓ) est une signature valide du message m, alors que (m, u^ℓ) n'a pas été demandé à $\mathcal{H}$. Puisque $\mathcal{H}(m, u^\ell)$ est uniformément distribué, l'égalité $M = \mathcal{H}(m, u^\ell)$ se produit avec une probabilité au plus 2^{-k}. Ainsi

$$|\Pr[\mathsf{Guess}_3] - \Pr[\mathsf{Guess}_2]| \leq \frac{q_\Upsilon}{2^k}.$$

Jeu$_5$ Finalement, dans la génération de la signature challenge, $\mathcal{D}$ tire un bit $b \in \{0,1\}$ au hasard, ainsi qu'un triplet $(Q_A^\star, Q_B^\star, \ell^\star) \in \mathbb{G}^2 \times [\![1, q-1]\!]$, qu'il propose en guise de signature challenge sur $m^\star$, mais sans mettre à jour la H-List. Ce jeu final est indistinguable du jeu précédent à moins que (m, v), ou $v = u^{\ell^\star}$, ait été soumis à $\mathcal{H}$ par l'oracle de signature, de vérification ou par l'attaquant. Le premier cas a été écarté dans le jeu numéro 3, et par définition du scénario PSI-CMA, le dernier ne peut pas se produire (la vérification se ferait sur la signature challenge). Ainsi, puisque $u = v^{1/\ell^\star}$ ou bien $u = v^{1/\alpha\ell^\star}$,

$$|\Pr[\mathsf{Guess}_5] - \Pr[\mathsf{Guess}_4]| \leq \varepsilon'.$$

Puisque la signature challenge n'apporte aucune information à propos de b à $\mathcal{A}$, on a de plus

$$\Pr[\mathsf{Guess}_5] = \frac{1}{2},$$

ce qui termine la preuve.

□

Remarque 12 *Dans la simulation, il est possible de s'arrêter dès que l'oracle DBDH répond* valid *lorsqu'on l'interroge sur un quadruplet $(X, Y, Z, v^{1/\ell})$, puisqu'alors, $v^{1/\ell}$ est une solution à notre instance du problème GBDH.*

Remarque 13 *La sécurité PSI-CMA se réduit en fait à une version plus faible du problème GBDH, puisqu'en fait, un oracle répondant à des instances du type $(X, Y, Z, \zeta) \in \mathbb{G}^3 \times \mathbb{H}$, avec X, Y, Z fixés est suffisant pour prouver cette sécurité.*

4.3.3 SMDVS : anonymat sans chiffrement

Il semble assez difficile, sans un chiffrement de la signature à vérificateurs désignés, d'obtenir la propriété d'anonymat du signataire. Dans cette section, nous proposons un schéma qui possède cet anonymat sans chiffrement, mais ce, au prix d'une taille de signature qui croît linéairement avec le nombre de vérificateurs désignés. Le schéma reste malgré cela efficace et spontané. Cette construction a fait l'objet d'une soumission de [146] avec D. Vergnaud.

4.3.3.1 Description

Le schéma SMDVS, décrit dans la figure 4.7, repose sur la même contruction que B2DVS, c'est à dire qu'il s'appuie sur un schéma de signature d'anneau, et sur un échange de clés entre n entités. Pour avoir un schéma non-interactif, c'est Alice qui calcule la “clé d'anonymat”, liée à un aléa connu d'elle seule, qu'elle suggère aux vérificateurs en le masquant dans la clé publique de ceux-ci. Cette clé n'est censée pouvoir se déduire de la signature que si l'on est l'un des vérificateurs, qui démasque l'aléa grâce à leur clé secrète. Seule la connaissance de cette clé permet de lever l'anonymat du signataire.

4.3.3.2 Sécurité

Consistance et source hiding Ces propriétés sont vraies pour les mêmes raisons que dans le schéma B2DVS. La clé d'anonymat peut être produite par n'importe lequel des vérificateurs, puis, la collaboration de tous leur permet de produire des signatures inconditionnellement analogues à celle d'Alice.

Génération des paramètres publics : SMDVS.Setup

• k est un paramètre de sécurité
• $(q, P, \mathbb{G}, \mathbb{H}, e)$ est généré par BDHGen(k)
• Soit $[\{0,1\}^* \times \mathbb{G}^{n+2} \longrightarrow \mathbb{G}]$ une famille de fonctions de hachage H un élément pris au hasard dans cette famille.

Génération des clés du signataire : SMDVS.SKeyGen

SMDVS.SKeyGen = B2DVS.SKeyGen

Génération des clés du vérificateur : SMDVS.VKeyGen

SMDVS.VKeyGen = B2DVS.VKeyGen

Signature à vérificateurs désignés : SMDVS.Sign

• Étant donné un message m, la clé secrète de signature a, les clés publiques P_{B_i} pour $i \in [\![1, n]\!]$ des vérificateurs désignés,
• Alice calcule la “clé d'anonymat” en tirant au hasard $\ell \in [\![1, q-1]\!]$ et en posant, pour tout $i \in [\![1, n]\!]$ $Y_{B_i} = \ell P_{B_i}$, et $Y = \ell P$.
• Alice hache alors le message, les clés publiques et la clé d'anonymat de la façon suivante : $M = H(m, P_A, P_{B_1}, \ldots, P_{B_n}, Y)$.
• Finalement, elle tire un deuxième aléa $r \in [\![1, q-1]\!]$ et pose $P_B = P_{B_1} + \cdots + P_{B_n}$ puis

$$Q_A = a^{-1}(M - rP_B) \text{ et } Q_B = rP.$$

• La signature de m est le $n+2$-uplet $\sigma = (Q_A, Q_B, Y_{B_1}, \ldots, Y_{B_n})$.

Verification désignée : SMDVS.Verify

• Étant donné une paire (m, σ), la clé publique de signature P_A, chaque B_i retrouve Y en calculant $b_i^{-1} Y_{B_i} = \ell P$.
• Le vérificateur B_i vérifie, pour tout $j \in [\![1, n]\!] \setminus \{i\}$, que $e(P_{B_j}, \ell P) = e(Y_{B_j}, P)$.
• Il calcule alors $M = H(m, P_A, P_{B_1}, \ldots, P_{B_n}, Y)$ et accepte la signature si

$$e(Q_A, P_A) e(Q_B, P_B) = e(M, P).$$

Fig. 4.7 – SMDVS : description

Inforgeabilité Le théorème suivant montre que l'existence d'un attaquant EF-CMA contre SMDVSpermet de construire un algorithme qui résout le problème CDH. La réduction n'est pas particulièrement bonne puisqu'il faut parier sur l'un des n vérificateurs pour injecter une information exploitable dans la preuve liée à l'instance CDH, et qu'en fait il faut répéter deux fois l'attaque pour résoudre CDH (comme pour B2DVS, l'attaquant EF-CMA permet de résoudre le problème CDH').

Théorème 29 (Inforgeabilité de SMDVS) *Soit BDHGen un générateur de groupe BDH symétrique et SMDVS le schéma de signature à vérificateurs désignés associés. Pour tout adversaire EF-CMA $\mathcal{A}$, dans le modèle de l'oracle aléatoire, contre SMDVS de paramètre de sécurité k, ayant une probabilité de succès $\varepsilon = \mathbf{Succ}^{\mathsf{ef-cma}}_{\mathsf{SMDVS},\mathcal{A}}$, en temps τ, après $q_{\mathcal{H}}$ requêtes à l'oracle aléatoire, q_Σ requêtes aux oracles de signatures et q_Υ requêtes à l'oracle de vérification, il existe un adversaire D contre CDH, ayant un succès $\varepsilon' = \mathbf{Succ}^{\mathsf{cdh}}_{\mathsf{BDHGen},D}(k)$ en temps $\tau' \in \mathbb{N}$ avec*

$$\left\{\begin{array}{lcl} \varepsilon' & \geq & \left(\dfrac{1}{n}\varepsilon - \dfrac{q_{\mathcal{H}} q_\Sigma + 1}{2^k}\right)^2 \\ \tau' & \leq & 2\left(\tau + (q_{\mathcal{H}} + 2q_\Sigma + O(1))T_{\mathbb{G}} + q_\Sigma T_{\mathbb{H}}\right) \end{array}\right.$$

Preuve : Elle se déduit immédiatement de la preuve du théorème 27 sur l'inforgeabilité de B2DVS. □

Anonymat L'anonymat de SMDVS repose aussi sur le problème CDH. La réduction est donnée dans le théorème suivant.

Théorème 30 (Anonymat de SMDVS) *Soit BDHGen un générateur de groupe BDH symétrique et SMDVS le schéma de signature à vérificateurs désignés associés. Pour tout adversaire PSI-CMA $\mathcal{A}$, dans le modèle de l'oracle aléatoire, contre SMDVS de paramètre de sécurité k, ayant avantage $\varepsilon = \mathbf{Adv}^{\mathsf{psi-cma}}_{\mathsf{SMDVS},\mathcal{A}}$, en temps τ, après $q_{\mathcal{H}}$ requêtes à l'oracle aléatoire, q_Σ requêtes aux oracles de signatures et q_Υ requêtes à l'oracle de vérification, il existe un adversaire D contre CDH, ayant un succès $\varepsilon' = \mathbf{Succ}^{\mathsf{cdh}}_{\mathsf{BDHGen},D}(k)$ en temps $\tau' \in \mathbb{N}$ avec*

$$\left\{\begin{array}{lcl} \varepsilon' & \geq & \left(\dfrac{\varepsilon}{2} - \dfrac{q_\Sigma + q_\Upsilon}{2^k}\right)^2 \\ \tau' & \leq & 2\tau + 2(q_{\mathcal{H}} + O(1))(q_\Upsilon + O(1))T_P \end{array}\right.$$

où T_P est le temps d'évaluation d'un couplage.

Preuve :

Soit k un paramètre de sécurité, et BDHGen un générateur de groupes BDH. On note $(q, P, \mathbb{G}, \mathbb{H}, e) = \mathsf{BDHGen}(k)$. Considérons l'instance aléatoire du problème CDH', déjà rencontré dans la preuve d'inforgeabilité de B2DVS (théorème 27) suivante : $R_x = xP$ et $R_{xy} = xyP$ pour $(x, y) \in [\![1, q-1]\!]$. Grâce à l'attaquant PSI-CMA $\mathcal{A}$ contre SMDVS, nous allons construire un algorithme $\tilde{\mathcal{D}}$ qui calcule le point $R_y = yP$.

Jeu_0 Dans ce jeu qui est en fait le scénario de l'attaque réelle, deux paires de clés de signatures sont produites par l'algorithme de génération de clés pour le signataire : on les note (P_{A_0}, a_0) et (P_{A_1}, a_1). L'algorithme de génération de clés pour les vérificateurs produit n paires de clés (P_{B_i}, b_i) pour chaque B_i, pour tout $i \in [\![1, n]\!]$. L'adversaire PSI-CMA contre SMDVS $\mathcal{A} = (\mathsf{find}, \mathsf{guess})$ reçoit les $n+2$ clés publiques générées, et, à l'issue de sa première étape find, propose au challenger un message $m^\star$. Le challenger tire alors un bit $b \in \{0, 1\}$ au hasard, et produit la signature $\sigma^\star = \mathsf{SMDVS.Sign}(m^\star, a_b, P_{B_1}, \ldots, P_{B_n})$. Cette dernière est ensuite donnée à $\mathcal{A}$ qui, à la fin de sa deuxième phase guess, sort un bit $b^\star$. Durant ses deux

phases, l'attaquant a un accès permanent à l'oracle aléatoire $\mathcal{H}$, aux oracles de signatures Σ_0 et Σ_1, et à l'oracle de vérification Υ. On note $q_{\mathcal{G}}$, q_{Σ_0}, q_{Σ_1} et q_{Υ} le nombre de requêtes aux différents oracles, et $q_{\Sigma} = q_{\Sigma_0} + q_{\Sigma_1}$.
On note Guess_0 l'événement $b^\star = b$, et on fait de même pour l'événement Guess_i dans le Jeu_i.

Par définition,

$$|2\Pr[\mathsf{Guess}_0] - 1| = \mathbf{Adv}^{\mathsf{psi-cma}}_{\mathsf{SMDVS},\mathcal{A}}(k)$$

Jeu_1 Dans ce jeu, l'algorithme $\tilde{\mathcal{D}}$ tire au hasard $\tilde{a_0}$ et $\tilde{a_1}$ dans $[\![1, q-1]\!]$, et donne à l'attaquant les nouvelles clés $P_{A_0} = \tilde{a_0}P$ et $P_{A_1} = \tilde{a_1}P$. Il tire ensuite au hasard n entiers $s_i \in [\![1, q-1]\!]$ pour $i \in [\![1, n]\!]$ et donne à l'attaquant comme clés publiques des vérificateurs désignés les points $P_{B_i} = s_i R_x$, pour $i \in [\![1, n]\!]$. Cette simulation est bien sûr indistinguable du jeu réel :

$$\Pr[\mathsf{Guess}_1] = \Pr[\mathsf{Guess}_0].$$

Jeu_2 L'oracle aléatoire $\mathcal{H}$ est alors simulé par $\tilde{\mathcal{D}}$ en maintenant à jour une liste de hachés $\mathsf{H\text{-}List}$ de façon usuelle.

$$\Pr[\mathsf{Guess}_2] = \Pr[\mathsf{Guess}_1].$$

Jeu_3 La connaissance des clés secrètes $\tilde{a_0}$ et $\tilde{a_1}$ permet à $\tilde{\mathcal{D}}$ de parfaitement simuler les oracles de signature.

$$\Pr[\mathsf{Guess}_3] = \Pr[\mathsf{Guess}_2].$$

Jeu_4 Dans ce jeu, l'algorithme $\tilde{\mathcal{D}}$ impose à l'oracle de vérification de rejeter tous les triplets message/signature/entité $(m, (Q_A, Q_B, Y_{B_1}, \ldots, Y_{B_n}), A_i)$ pour lesquels le message m n'a pas été préalablement proposé à l'oracle de hachage. La simulation diffère de la précédente si $(Q_A, Q_B, Y_{B_1}, \ldots, Y_{B_n})$ est une signature valide de m par A_i alors que $\mathcal{H}(m, P_A, P_{B_1}, \ldots, P_{B_n}, Y)$ n'a pas été demandé à l'oracle $\mathcal{H}$. L'égalité $\mathcal{H}(m, P_A, P_{B_1}, \ldots, P_{B_n}, Y) = \tilde{a}_i Q_A + r P_B$ se produit avec une probabilité au plus 2^{-k}. Ainsi

$$|\Pr[\mathsf{Guess}_4] - \Pr[\mathsf{Guess}_3]| \leq \frac{q_{\Upsilon}}{2^k}.$$

Jeu_5 $\tilde{D}$ peut alors simuler l'oracle de vérification de la façon suivante. Pour toute requête du type message/signature/entité $(m, (Q_A, Q_B, Y_{B_1}, \ldots, Y_{B_n}), A_i)$ à Υ, il parcourt la $\mathsf{H\text{-}List}$ et, pour chaque $(n+2)$-uplet de la forme $(m, P_{B_1}, \ldots, P_{B_n}, R)$, il teste si $e(R, P_{B_1}) = e(Y_{B_1}, P)$. Si tel est le cas, il vérifie, pour tous les $j \in [\![2, n]\!]$, que $e(R, P_{B_j}) = e(Y_{B_j}, P)$. Il calcule alors $M = H(m, P_A, P_{B_1}, \ldots, P_{B_n}, R)$ et le nouvel oracle accepte la signature si et seulement si $e(M, P) = e(Q_A, P_A)e(Q_B, P_B)$. Ce jeu simule parfaitement l'oracle de vérification :

$$\Pr[\mathsf{Guess}_5] = \Pr[\mathsf{Guess}_4].$$

Jeu_6 Il reste à $\tilde{\mathcal{D}}$ la simulation de la signature challenge. Il commence donc par tirer au hasard un bit $b \in \{0, 1\}$, puis

– il calcule $Y^\star_{B_i} = s_i R_{xy}$ pour $i \in [\![1, n]\!]$
– il tire au hasard $M^\star \in \mathbb{G}$ ainsi que $r^\star \in [\![1, q-1]\!]$,
– et calcule $Q^\star_B = r^\star P$ et $Q^\star_A = \tilde{a_b}^{-1}(M^\star - r^\star P_B)$.

Le challenger simulé sort la signature $\sigma^\star = (Q_A^\star, Q_B^\star, Y_{B_1}^\star, \ldots, Y_{B_n}^\star)$ sans mettre à jour la H-List avec $M^\star$.

À la fin de ce jeu, l'adversaire $\mathcal{A}$ propose un bit $b^\star$. $\tilde{\mathcal{D}}$ parcourt alors la H-List, à la recherche des $(n+2)$-uplets de la forme $(m^\star, P_{B_1}, \ldots, P_{B_n}, R)$ et il teste si $e(R, R_x) = e(R_{xy}, P)$. Si c'est le cas, $\tilde{\mathcal{D}}$ ressort R, sinon, il renvoie un élément au hasard dans $\mathbb{G}$. Si l'on note $\tilde{\tau}$ et $\tilde{\varepsilon}$ le temps d'exécution de $\tilde{\mathcal{D}}$ son succès pour résoudre CDH', alors $\tilde{\tau}$ est majoré par $\tau + (q_{\mathcal{H}} + O(1))(q_\Upsilon + O(1))T_P$ où T_P est le temps d'évaluation d'un couplage.
Cette simulation est indistinguable de la précédente, à moins que $(m^\star, P_{B_1}, \ldots, P_{B_n}, Y^\star)$ où $Y^\star = yP$ ait été demandé à $\mathcal{H}$ par l'oracle de signature, l'oracle de vérification ou l'adversaire. Le premier cas se produit avec une probabilité au plus $q_\Sigma 2^{-k}$, et par définition de la sécurité PSI-CMA le second cas ne peut pas se produire : il est interdit de demander la vérification de la signature challenge. La probabilité que $(m^\star, P_{B_1}, \ldots, P_{B_n}, Y^\star)$ soit demandé à $\mathcal{H}$ par l'adversaire est majorée par le succès de $\tilde{\mathcal{D}}$, $\tilde{\varepsilon}$. La signature challenge ne donne aucune information à $\mathcal{A}$ à propos de b si $(m^\star, P_{B_1}, \ldots, P_{B_n}, Y^\star)$ n'est pas demandé à $\mathcal{H}$, et ainsi, il est victorieux avec probabilité 1/2.

Ainsi, on obtient

$$\begin{aligned}
\frac{\varepsilon}{2} &= \left| \frac{\mathbf{Adv}^{\mathsf{psi-cma}}_{\mathsf{SMDVS},\mathcal{A}}(k) + 1}{2} - \frac{1}{2} \right| \\
&= \left| \Pr[\mathcal{A} \text{ gagne dans le jeu réel}] - \Pr[\mathcal{A} \text{ gagne dans la simulation de } \tilde{\mathcal{D}}] \right| \\
&\leq \tilde{\varepsilon} + \frac{q_\Sigma + q_\Upsilon}{2^k}.
\end{aligned}$$

□

4.3.3.3 Remarque

La propriété d'anonymat du signataire est difficile à obtenir pour les signatures à vérificateur désigné. Elle est hautement liée à un échange de clé : dans SMDVS, c'est $Y = \ell P$ qui fait office de clé d'anonymat, et qui est partagée entre le signataire et les vérificateurs. L'ingrédient dont nous avions besoin était un échange de clé qui pouvait être initié par le signataire, et qui ne nécessitait pas d'interactions entre les vérificateurs. Dans le cas de SMDVS, cet échange multi-partite est très simple : Alice masque le logarithme discret de cette clé dans les clés publiques des vérificateurs. C'est aussi l'échange de clé tripartite de Joux qui permet d'obtenir l'anonymat du signataire dans le schéma B2DVS. Et pour ce qui est de DVSBMH, la clé abP est aussi implicitement utilisée.

4.3.4 Conclusion

Nous avons défini les signatures à vérificateurs désignés, qui généralisent les signatures de Jakobsson *et al.* et développé la propriété d'anonymat du signataire, qui semble importante en pratique, dans des contextes où l'anonymat est nécessaire. Nous avons proposé un schéma efficace de signature à simple vérificateur, puis discuté des constructions possibles pour plusieurs vérificateurs. Ce travail a été étendu par Chow dans [60] qui prône l'utilisation de fonctions caméléons dans le schéma basé sur les signatures d'anneau. C'est une remarque intéressante, cependant, Chow justifie leur utilisation pour se passer du calcul multi-partite du côté des vérificateurs, or ces derniers ne mettent en réalité jamais en place ce calcul.

Les signatures à vérificateurs désignés ont la propriété de convaincre les membres du groupe formé par les vérificateurs, mais pas une personne extérieure au groupe. Cependant, celle-ci peut avoir la conviction que cette signature a été produite soit par Alice, soit par les vérificateurs tous ensemble. Or, il pourrait être intéressant que des signatures aient la propriété de convaincre tous les membres d'un groupe, au sens où tous seraient convaincus qu'une signature a été produite par un membre donné du groupe, alors que toute autre personne ne serait convaincue que du fait que la signature provient d'un membre du groupe. Les signatures à vérificateurs désignés ne sont pas tout à fait satisfaisantes, puisqu'une personne extérieure au groupe devrait avoir une probabilité égale à $1/n$ de deviner le signataire, où n est le nombre de membres du groupe. Ces signatures devraient être une variante des signatures d'anneau ou des signatures de groupes.

Il pourrait être intéressant d'avoir une propriété de conversion des signatures à vérificateurs désignés convertibles, c'est-à-dire de donner la possibilité au signataire de lever l'anonymat. Dans le schéma B2DVS, Alice pourrait par exemple faire une preuve à vérificateur désigné de sa connaissance du r dans le point Q_{BC}. Elle peut ensuite rendre cette preuve non interactive, et ainsi la signature devient universellement vérifiable. Cette conversion n'est pas possible dans le schéma DVSBMH : l'ambiguïté du signataire inconditionnelle l'empêche définitivement dans ce cas là.

5

Conclusion

Dans cette thèse, nous avons contribué à la conception de schémas de signatures “spéciales”. Les signatures à vérification contrôlée constituent le cœur de cette étude. Elles sont essentiellement toutes des variantes des signatures indéniables de Chaum et van Antwerpen.

Nous avons enrichi la définition de ces dernières en introduisant le concept de conversion temporelle, puis proposé des nouveaux schémas dont la sécurité repose sur des variantes bilinéaires du problème Diffie-Hellman. Nous avons pu en particulier prouver la sécurité d'une variante indéniable des signatures de Boneh et Boyen dans le modèle standard, en considérant des groupes sans application bilinéaire admissible.

Une variante naturelle des signatures indéniables a ensuite été abordée : les signatures dirigées sont telles que le signataire peut déléguer sa capacité à confirmer ou démentir une signature. Le schéma que nous proposons est une variante naturelle du schéma de nos signatures indéniables. Sa sécurité repose sur une variante du problème xyz-DDH et sur une variante simple du problème Diffie-Hellman bilinéaire dans le modèle de l'oracle aléatoire.

Les signatures à vérificateurs désignés, que nous avons formellement définies dans un contexte multi-acteurs, ont fait l'objet de la dernière partie de ce mémoire. Une étude intensive a été conduite, à travers une amélioration notable du schéma de Steinfeld *et al.* et l'élaboration de schémas permettant au signataire de désigner plusieurs vérificateurs.

Dans tous ces schémas, nous nous sommes intéressés à protéger efficacement l'anonymat du signataire. L'utilisation des applications bilinéaires a grandement contribué à obtenir des solutions efficaces, et l'astuce xyz en particulier permet d'atteindre des compromis pertinents entre authenticité et anonymat. Le problème xyz-DDH semble promis à un avenir prospère dans la conception de schémas de signatures spéciales. L'étude de la sécurité de tels schémas est fondamentale et doit être faite de façon la plus rigoureuse possible, en évitant autant que possible les hypothèses idéales, comme celle de l'oracle aléatoire. Aussi, concevoir des schémas dont la sécurité repose sur des problèmes classiques ou positionnés de façon précise dans la hiérarchie de ces problèmes et puisse être prouvée dans le modèle standard, reste une piste de recherche largement ouverte. La variété des solutions mises au point permettra sans aucun doute d'élargir notablement le champ d'application des signatures spéciales, et ce, d'autant plus que les transactions électroniques ont des exigences de sécurité qui ne cessent de croître.

Bibliographie

[1] M. Abe, M. Ohkubo, and K. Suzuki. 1-out-of-n Signatures from a Variety of Keys. In *Advances in Cryptology - Proceedings of Asiacrypt'02*, volume 2501 of *Lecture Notes in Computer Science*, pages 415–432. Springer-Verlag, 2002.

[2] L. M. Adleman. A Subexponential Algorithm for the Discrete Logarithm Problem with Applications to Cryptography. In *Proceedings of the 20th IEEE Symposium on Foundations of Computer Science - FOCS*, pages 55–60. IEEE, 1979.

[3] L. M. Adleman and J. De Marrais. A Subexponential Algorithm for Discrete Logarithms over All Finite Fields. *Mathematics of Computations*, 61(203) :1–15, 1993. Also in Proceedings of Crypto'93, Springer LNCS Vol. 773, 147-158 (1993).

[4] S. Araki, S. Uehara, and K. Imamura. The Limited Verifier Signature and Its Application. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E82-A(1) :63–68, 1999.

[5] N. Asokan, V. Shoup, and M. Waidner. Optimistic Fair Exchange of Digital Signatures. In *Advances in Cryptology - Proceedings of Eurocrypt'98*, volume 1403 of *Lecture Notes in Computer Science*, pages 591–606. Springer-Verlag, 1998.

[6] A. O. L. Atkin and F. Morain. Elliptic Curves and Primality Proving. *Mathematics of Computations*, pages 29–68, 1993.

[7] R. Balasubramanian and N. Koblitz. The Improbability that an Elliptic Curve Has Subexponential Discrete Log Problem under the Menezes-Okamoto-Vanstone Algorithm. *Journal of Cryptology*, 11(2) :141–145, 1998.

[8] N. Baric and B. Pfitzmann. Collision-Free Accumulators and Fail-Stop Signature Schemes Without Trees. In *Advances in Cryptology - Proceedings of Eurocrypt'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 480–494. Springer-Verlag, 1997.

[9] P. S. L. M. Barreto, S. Galbraith, C. O'hEigeartaigh, and M. Scott. Efficient Pairing Computation on Supersingular Abelian Varieties. Cryptology ePrint Archive, Report 2004/375, 2004. `http://eprint.iacr.org/`.

[10] P. S. L. M. Barreto and H. Y. Kim. Fast Hashing onto Elliptic Curves over Fields of Characteristic 3. Cryptology ePrint Archive, Report 2001/098, 2001. `http://eprint.iacr.org/`.

[11] P. S. L. M. Barreto, H. Y. Kim, B. Lynn, and M. Scott. Efficient Algorithms for Pairing-Based Cryptosystems. In *Advances in Cryptology - Proceedings of Crypto'02*, volume 2442 of *Lecture Notes in Computer Science*, pages 354–368. Springer-Verlag, 2002.

[12] P. S. L. M. Barreto, B. Lynn, and M. Scott. Efficient Implementation of Pairing-Based Cryptosystems. *Journal of Cryptology*, 17(4) :312–334, 2004.

[13] M. Bellare. Practice-Oriented Provable-Security. In *Proceedings of First International Workshop on Information Security (ISW 97)*, volume 1396 of *Lecture Notes in Computer Science*, pages 221–231. Springer-Verlag, 1998. Invited Lecture.

[14] M. Bellare, A. Boldyreva, A. Desai, and D. Pointcheval. Key-Privacy in Public-Key Encryption. In *Advances in Cryptology - Proceedings of Asiacrypt'01*, volume 2248 of *Lecture Notes in Computer Science*, pages 566–582. Springer-Verlag, 2001.

[15] M. Bellare, A. Boldyreva, and A. Palacio. An Uninstantiable Random-Oracle-Model Scheme for a Hybrid-Encryption Problem. In *Advances in Cryptology - Proceedings of Eurocrypt'04*, volume 3027 of *Lecture Notes in Computer Science*, pages 171–188. Springer-Verlag, 2004.

[16] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among Notions of Security for Public-Key Encryption Schemes. In *Advances in Cryptology - Proceedings of Crypto'98*, volume 1462 of *Lecture Notes in Computer Science*, pages 162–177. Springer-Verlag, 1998.

[17] M. Bellare, J. Kilian, and P. Rogaway. The Security of Cipher Block Chaining. In *Advances in Cryptology - Proceedings of Crypto'94*, volume 839 of *Lecture Notes in Computer Science*, pages 341–358. Springer-Verlag, 1994.

[18] M. Bellare, J. Kilian, and P. Rogaway. The Security of the Cipher Block Chaining Message Authentication Code. *Journal of Computer and System Sciences*, 61(3) :362–399, 2000.

[19] M. Bellare and S. Micali. How to Sign Given Any Trapdoor Permutation. *Journal of the ACM*, 39(1) :214–233, 1992.

[20] M. Bellare and C. Namprempre. Authenticated Encryption : Relations Among Notions and Analysis of the Generic Composition Paradigm. In *Advances in Cryptology - Proceedings of Asiacrypt'00*, volume 1976 of *Lecture Notes in Computer Science*, pages 317–330. Springer-Verlag, 2000.

[21] M. Bellare and A. Palacio. GQ and Schnorr Identification Schemes : Proofs of Security against Impersonation under Active and Concurrent Attacks. In *Advances in Cryptology - Proceedings of Crypto'02*, volume 2442 of *Lecture Notes in Computer Science*, pages 162–177. Springer-Verlag, 2002.

[22] M. Bellare and A. Palacio. The Knowledge-of-Exponent Assumptions and 3-Round Zero-Knowledge Protocols. In *Advances in Cryptology - Proceedings of Crypto'04*, volume 3152 of *Lecture Notes in Computer Science*. Springer-Verlag, 2004.

[23] M. Bellare and P. Rogaway. Random Oracles are Practical : a Paradigm for Designing Efficient Protocols. In *Proceedings of the First ACM Conference on Computer and Communications Security*, pages 62–73. ACM Press, 1993.

[24] M. Bellare and P. Rogaway. Optimal Asymmetric Encryption. In *Advances in Cryptology - Proceedings of Eurocrypt'94*, volume 950 of *Lecture Notes in Computer Science*, pages 92–111. Springer-Verlag, 1995.

[25] M. Bellare and P. Rogaway. The Exact Security of Digital Signatures : How to Sign with RSA and Rabin. In *Advances in Cryptology - Proceedings of Eurocrypt'96*, volume 1070 of *Lecture Notes in Computer Science*, pages 399–416. Springer-Verlag, 1996.

[26] M. Bellare and A. Sahai. Non-Malleable Encryption : Equivalence between Two Notions, and an Indistinguishability-Based Characterization. In *Advances in Cryptology -*

Proceedings of Crypto'99, volume 1666 of *Lecture Notes in Computer Science*, pages 519–536. Springer Verlag, 1999.

[27] I. Biehl, S. Paulus, and T. Takagi. Efficient Undeniable Signature Schemes Based on Ideal Arithmetic in Quadratic Orders. *Designs, Codes and Cryptography*, 31(2) :99–123, 2004.

[28] I. Blake, G. Seroussi, and N. Smart. *Elliptic Curves in Cryptography*. London Mathematical Society Lecture Note Series. Cambridge University Press, 1999.

[29] M. Blum, P. Feldman, and S. Micali. Non-Interactive Zero-Knowledge and Its Applications. In *Proceedings of the 20rd ACM Symposium on Theory of Computing STOC'88*, pages 103–112. ACM Press, 1988.

[30] M. Blum and S. Goldwasser. An Efficient Probabilistic Public-Key Encryption Scheme Which Hides All Partial Information. In *Advances in Cryptology - Proceedings of Crypto'84*, volume 196 of *Lecture Notes in Computer Science*, pages 289–302. Springer-Verlag, 1985.

[31] M. Blum and S. Micali. How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits. *SIAM Journal on Computing*, 13(4) :850–864, 1984.

[32] D. Boneh. The Decision Diffie-Hellman Problem. In *Proceedings of the Third Algorithmic Number Theory Symposium*, volume 1423 of *Lecture Notes in Computer Science*, pages 48–63. Springer-Verlag, 1998. Invited Talk.

[33] D. Boneh. Simplified OAEP for the RSA and Rabin functions. In *Advances in Cryptology - Proceedings of Crypto'01*, volume 2139, pages 275–291. Springer Verlag, 2001.

[34] D. Boneh and X. Boyen. Short Signatures Without Random Oracles. In *Advances in Cryptology - Proceedings of Eurocrypt'04*, volume 3027 of *Lecture Notes in Computer Science*, pages 56–73. Springer-Verlag, 2004.

[35] D. Boneh and M. Franklin. Identity-Based Encryption from the Weil Pairing. In *Advances in Cryptology - Proceedings of Crypto'01*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer-Verlag, 2001.

[36] D. Boneh and M. Franklin. Identity-Based Encryption from the Weil Pairing. *SIAM Journal of Computing*, 32(3) :586–615, 2003.

[37] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and Verifiably Encrypted Signatures from Bilinear Maps. In *Advances in Cryptology - Proceedings of Eurocrypt'03*, volume 2656 of *Lecture Notes in Computer Science*, pages 416–432. Springer-Verlag, 2003.

[38] D. Boneh, B. Lynn, and H. Shacham. Short Signatures from the Weil Pairing. In *Advances in Cryptology - Proceedings of Asiacrypt'01*, volume 2248 of *Lecture Notes in Computer Science*, pages 514–532. Springer-Verlag, 2001.

[39] D. Boneh, B. Lynn, and H. Shacham. Short Signatures from the Weil Pairing. *Journal of Cryptology*, 17(4) :297–319, 2004.

[40] D. Boneh, I. Mironov, and V. Shoup. A Secure Signature Scheme from Bilinear Maps. In *Topics in Cryptology - Proceedings of CT-RSA'03*, volume 2612 of *Lecture Notes in Computer Science*, pages 98–110. Springer-Verlag, 2003.

[41] D. Boneh and R. Venkatesan. Breaking RSA May Not Be Equivalent to Factoring. In *Advances in Cryptology - Proceedings of Eurocrypt'98*, volume 1403 of *Lecture Notes in Computer Science*, pages 59–71. Springer-Verlag, 1998.

[42] J. Boyar, D. Chaum , I. B. Damgård, and T.P. Pedersen. Convertible Undeniable Signatures. In *Advances in Cryptology - Proceedings of Crypto'90*, Lecture Notes in Computer Science, pages 189–205. Springer-Verlag, 1991.

[43] G. Brassard, D. Chaum, and C. Crépeau. Minimum Disclosure Proofs of Knowledge. *Journal of Computer and System Sciences*, 37(2) :156–189, 1988.

[44] E. Bresson. *Protocoles cryptographiques pour l'authentification et l'anonymat dans les groupes.* PhD thesis, École polytechnique, 2002.

[45] E. Bresson, J. Stern, and M. Szydlo. Threshold Ring Signatures and Applications to Ad-hoc Groups. In *Advances in Cryptology - Proceedings of Crypto'02*, volume 2442 of *Lecture Notes in Computer Science*, pages 465–480. Springer-Verlag, 2002.

[46] D. R. L. Brown and R. P. Gallant. The Static Diffie-Hellman Problem. Cryptology ePrint Archive, Report 2004/306, 2004. `http://eprint.iacr.org/`.

[47] J. Camenisch. Efficient and Generalized Group Signatures. In *Advances in Cryptology - Proceedings of Eurocrypt'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 465–479. Springer-Verlag, 1997.

[48] J. Camenisch and I. Damgård. Verifiable Encryption, Group Encryption, and Their Applications to Separable Group Signatures and Signature Sharing Schemes. In *Advances in Cryptology - Proceedings of Asiacrypt'00*, volume 1976 of *Lecture Notes in Computer Science*, pages 331–345. Springer-Verlag, 2000.

[49] J. Camenisch and M. Michels. Confirmer Signature Schemes Secure against Adaptative Adversaries. In *Advances in Cryptology - Proceedings of Eurocrypt'00*, volume 1807 of *Lecture Notes in Computer Science*, pages 243–258. Springer-Verlag, 2000.

[50] J. Camenisch and M. Stadler. Efficient Group Signature Schemes for Large Groups. In *Advances in Cryptology - Proceedings of Crypto'97*, volume 1294 of *Lecture Notes in Computer Science*, pages 410–424. Springer-Verlag, 1997.

[51] S. Canard. *Signatures de groupe, variantes et applications.* PhD thesis, Université de Caen, 2003.

[52] R. Canetti, O. Goldreich, and S. Halevi. The Random Oracles Methodology, Revisited. In *Proceedings of the 30th annual ACM symposium on Theory of computing*, pages 209–218. ACM Press, 1998.

[53] R. Canetti, O. Goldreich, and S. Halevi. The Random Oracles Methodology, Revisited. *Journal of the ACM*, 51(4) :557–594, 2004.

[54] D. Chaum. Zero-Knowledge Undeniable Signatures. In *Advances in Cryptology - Proceedings of Eurocrypt'90*, volume 473 of *Lecture Notes in Computer Science*, pages 458–464. Springer-Verlag, 1991.

[55] D. Chaum. Designated Confirmer Signatures. In *Advances in Cryptology - Proceedings of Eurocrypt'94*, volume 950 of *Lecture Notes in Computer Science*, pages 86–91. Springer-Verlag, 1995.

[56] D. Chaum. Private Signature and Proof Systems, 1996. U.S. Patent 5,493,614.

[57] D. Chaum and H. van Antwerpen. Undeniables signatures. In *Advances in Cryptology - Proceedings of Crypto'89*, volume 435 of *Lecture Notes in Computer Science*, pages 212–216, 1990.

[58] D. Chaum and E. van Heyst. Group Signatures. In *Advances in Cryptology - Proceedings of Eurocrypt'91*, volume 547 of *Lecture Notes in Computer Science*, pages 257–265. Springer-Verlag, 1991.

[59] X. Chen, F. Zhang, and K. Kim. Limited Verifier Signature from Bilinear Pairings. In *Applied Cryptography and Network Security - Proceedings of ACNS'04*, volume 3089 of *Lecture Notes in Computer Science*, pages 135–148. Springer-Verlag, 2004.

[60] S.S.M. Chow. Generic Multi-Designated Verifiers Signature Scheme. Submitted.

[61] M. Ciet. *Aspects of Fast and Secure Arithmetics for Elliptic Curve Cryptography.* PhD thesis, Université catholique de Louvain, 2003.

[62] D. Coppersmith. Fast Evaluation of Logarithms in Fields of Characteristic Two. *IEEE Transactions on Information Theory*, 30(4) :587–594, 1978.

[63] J.-S. Coron. On the Exact Security of Full Domain Hash. In *Advances in Cryptology - Proceedings of Crypto'00*, volume 1880 of *Lecture Notes in Computer Science*, pages 229–235. Springer-Verlag, 2000.

[64] J.-S. Coron. Optimal security proofs for PSS and other signature schemes. In *Advances in Cryptology - Proceedings of Eurocrypt'02*, volume 2332 of *Lecture Notes in Computer Science*, pages 272–287. Springer-Verlag, 2002.

[65] R. Cramer and S. Shoup. A Practical Public Key Cryptosystem Provably Secure Against Adaptive Chosen Ciphertext Attack. In *Advances in Cryptology - Proceedings of Crypto'98*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25, 1998.

[66] R. Cramer and S. Shoup. Signature Schemes Based on the Strong RSA Assumption. In *Proceedings of the 6th ACM Conference on Computer and Communications Security*, pages 46–51. ACM Press, 1999. Full version in ACM TISSEC 3(3), 161–185 (2000).

[67] B. Daireaux, V. Maume-Deschamp, and B. Vallée. The Lyapunov Tortoise and the Dyadic Hare. *Discrete Mathematics and Theoretical Computer Science*, 2005.

[68] E. Dall'Olio and O. Markowitch. Voting with Designated Verifier Signature-like Protocol. In *IADIS International Conference WWW/Internet 2004*, pages 295–301. Iadis Press, 2004.

[69] I. B. Damgård. Towards Practical Public Key Systems Secure Against Chosen Ciphertext Attacks. In *Advances in Cryptology - Proceedings of Crypto'91*, volume 576 of *Lecture Notes in Computer Science*, pages 445–456. Springer-Verlag, 1992.

[70] I. B. Damgård and T.P. Pedersen. New Convertible Undeniable Signature Schemes. In *Advances in Cryptology - Proceedings of Eurocrypt'96*, volume 1070 of *Lecture Notes in Computer Science*, pages 372–386. Springer-Verlag, 1996.

[71] B. den Boer. Diffie-Hellman is as Strong as Discrete Log for Certain Primes. In *Advances in Cryptology - Proceedings of Crypto'88*, volume 403 of *Lecture Notes in Computer Science*, pages 530–539. Springer-Verlag, 1989.

[72] A. W. Dent. A Designer's Guide to KEMs. In *Proceedings of the 9th IMA Conference on Coding and Cryptography*, volume 2898 of *Lecture Notes in Mathematics*, pages 133–151. Springer Verlag, 2003.

[73] Y. Desmedt. Verifier-designated signatures. Crypto'03 rump session, 2003.

[74] Y. Desmedt, C. Goutier, and S. Bengio. Special Uses and Abuses of the Fiat-Shamir Passport Protocol. In *Advances in Cryptology - Proceedings of Crypto'87*, volume 293 of *Lecture Notes in Computer Science*, pages 21–39. Springer-Verlag, 1988.

[75] Y. Desmedt and M. Yung. Weaknesses with Undeniable Signature Schemes. In *Advances in Cryptology - Proceedings of Eurocrypt'91*, volume 547 of *Lecture Notes in Computer Science*, pages 205–220. Springer-Verlag, 1991.

[76] W. Diffie and M.E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6) :644–654, 1976.

[77] J. D. Dixon. Asymptotically Fast Factorization of Integers. *Mathematics of Computations*, 36 :255–260, 1981.

[78] Y. Dodis, A. Kiayias, A. Nicolosi, and V. Shoup. Anonymous Identification in Ad Hoc Groups. In *Advances in Cryptology - Proceedings of Eurocrypt'04*, volume 3027 of *Lecture Notes in Computer Science*, pages 609–626. Springer-Verlag, 2004.

[79] D. Dolev, C. Dwork, and M. Naor. Non-Malleable Cryptography. In *Proceedings of the 23rd ACM Symposium on Theory of Computing STOC'91*, pages 542–552. ACM Press, 1991.

[80] D. Dolev, C. Dwork, and M. Naor. Non-Malleable Cryptography. *SIAM Journal on Computing*, 30(2) :391–437, 2000.

[81] R. Dupont. Applications des couplages sur les courbes elliptiques en cryptologie. Master's thesis, École polytechnique, 2002.

[82] R. Dupont, A. Enge, and F. Morain. Building Curves with Arbitrary Small MOV Degree over Finite Prime Fields. *Journal of Cryptology*, To appear.

[83] C. Dwork, M. Naor, and A. Sahai. Concurrent Zero-Knowledge. In *Proceedings of the 30th annual ACM Symposium on Theory of Computing*, pages 409–418. ACM Press, 1998.

[84] T. El Gamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *IEEE Transactions on Information Theory*, 31(4) :469–472, 1985.

[85] A. Enge. *Elliptic Curves and Their Applications to Cryptography : An Introduction.* Kluwer Academic Publishers, 1999.

[86] U. Feige, A. Fiat, and A. Shamir. Zero-knowledge proofs of identity. *Journal of Cryptology*, 1(2) :77–95, 1988.

[87] U. Feige and A. Shamir. Witness Indistinguishable and Witness Hiding Protocols. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 416–426. ACM Press, 1990.

[88] A. Fiat and A. Shamir. How to Prove Yourself : Practical Solutions to Identification and Signature Problems. In *Advances in Cryptology - Proceedings of Crypto'86*, volume 263, pages 186–194. Springer-Verlag, 1987.

[89] S. Flon and R. Oyono. Fast Arithmetic on Jacobians of Picard Curves. In *Public Key Cryptography - PKC 2004*, Lecture Notes in Computer Science, pages 55–68. Springer-Verlag, 2004.

[90] M. K. Franklin and M. K. Reiter. Verifiable Secret Sharing. In *Advances in Cryptology - Proceedings of Eurocrypt'95*, volume 921 of *Lecture Notes in Computer Science*, pages 50–63. Springer-Verlag, 1995.

[91] G. Frey and H. G. Rück. A Remark Concerning m-Divisibility and the Discrete Logarithm in the Divisor Class Group of Curves. *Mathematics of computation*, 62 :865–874, 1994.

[92] A. Fujioka, T. Okamoto, and K. Ohta. Interactive Bi-Proof Systems and Undeniable Signature Schemes. In *Advances in Cryptology - Proceedings of Eurocrypt'91*, volume 547 of *Lecture Notes in Computer Science*, pages 243–256. Springer-Verlag, 1991.

[93] E. Fujisaki, T. Okamoto, D. Pointcheval, and J. Stern. RSA-OAEP is Secure under the RSA Assumption. *Journal of Cryptology*, 17(2) :81–104, 2004.

[94] S. Galbraith. Supersingular Curves in Cryptography. In *Advances in Cryptology - Proceedings of Asiacrypt'01*, volume 2248 of *Lecture Notes in Computer Science*, pages 495–513. Springer-Verlag, 2001.

[95] S. Galbraith and W. Mao. Invisibility and Anonymity of Undeniable and Confirmer Signatures. In *Topics in Cryptology - CT-RSA 2003*, volume 2612 of *Lecture Notes in Computer Science*, pages 80–97. Springer-Verlag, 2003.

[96] S. Galbraith, W. Mao, and K. G. Paterson. RSA-based undeniable signatures for general moduli. In *Topics in Cryptology - CT-RSA 2002*, volume 2271 of *Lecture Notes in Computer Science*, pages 200–217. Springer-Verlag, 2002.

[97] S. Galbraith and N. P. Smart. A Cryptographic Application of Weil Descent. In *Proceedings of Cryptography and Coding '99*, volume 1746 of *Lecture Notes in Computer Science*, pages 191–200. Springer-Verlag, 1999.

[98] P. Gaudry. *Algorithmique des courbes hyperelliptiques et applications à la cryptologie.* PhD thesis, École Polytechnique, 12 décembre 2000.

[99] P. Gaudry. Index Calculus for Abelian Varieties and the Elliptic Curve Discrete Logarithm Problem. preprint, 2005.

[100] R. Gennaro, S. Halevi, and T. Rabin. Secure Hash-and-Sign Signatures without the Random Oracle. In *Advances in Cryptology - Proceedings of Eurocrypt'99*, volume 1592 of *Lecture Notes in Computer Science*, pages 123–139. Springer-Verlag, 1999.

[101] R. Gennaro, H. Krawczyk, and T. Rabin. RSA-based Undeniable Signatures. In *Advances in Cryptology - Proceedings of Crypto'97*, volume 1294 of *Lecture Notes in Computer Science*, pages 132–149. Springer-Verlag, 1997.

[102] M. Girault. Self-Certified Public Keys. In *Advances in Cryptology - Proceedings of Eurocrypt'91*, volume 547 of *Lecture Notes in Computer Science*, pages 490–497. Springer-Verlag, 1991.

[103] M. Girault and J.-C. Pailles. On-Line / Off-Line RSA-Like. In *Proceedings of the International Workshop on Coding and Cryptography - WCC 2003*, 2003.

[104] M. Girault, G. Poupard, and J. Stern. On The Fly Authentication and Signature Generation based on Groups of Unknown Order. *Journal of Cryptology*, 2005.

[105] E.-J. Goh and S. Jarecki. A Signature Scheme as Secure as the Diffie-Hellman Problem. In *Advances in Cryptology - Proceedings of Eurocrypt'03*, volume 2656 of *Lecture Notes in Computer Science*, pages 401–415. Springer-Verlag, 2003.

[106] O. Goldreich. Zero-knowledge twenty years after its invention. Technical Report TR02-063, Electronic Colloquium on Computational Complexity, 2002.

[107] O. Goldreich and H. Krawczyk. On the Composition of Zero-Knowledge Proof Systems. In *Proceedings of the 17th International Colloquium on Automata, Languages and Programming (ICALP'90)*, volume 443 of *Lecture Notes in Computer Science*, pages 268–282. Springer-Verlag, 1990.

[108] O. Goldreich and H. Krawczyk. On the Composition of Zero-Knowledge Proof Systems. *SIAM Journal on Computing*, 25(1) :169–192, 1996. Full version of [107].

[109] O. Goldreich, S. Micali, and A. Wigderson. How to Prove all NP-Statements in Zero-Knowledge, and a Methodology of Cryptographic Protocol Design. In *Advances in Cryptology - Proceedings of Crypto'86*, volume 263 of *Lecture Notes in Computer Science*, pages 171–185. Springer-Verlag, 1987.

[110] S. Goldwasser and J. Kilian. Almost All Primes Can Be Quickly Certified. In *Proceedings of the 18th ACM Symposium on Theory of Computing STOC'86*, pages 316–329. ACM Press, 1986.

[111] S. Goldwasser and S. Micali. Probabilistic Encryption and How to Play Mental Poker Hiding All Partial Information. In *Proceedings of the 14th Annual ACM Symposium on the Theory of Computing*, pages 365–377. ACM Press, 1982.

[112] S. Goldwasser and S. Micali. Probabilistic Encryption. *Journal of Computer and System Sciences*, 28(2) :270–299, 1984. Full version of [111].

[113] S. Goldwasser, S. Micali, and C. Rackoff. The Knowledge Complexity of Interactive Proof Systems. In *Proceedings of the 17th ACM Symposium on Theory of Computing STOC'85*, pages 291–304. ACM Press, 1985.

[114] S. Goldwasser, S. Micali, and C. Rackoff. The Knowledge Complexity of Interactive Proof Systems. *SIAM Journal of Computing*, 18(1) :186–207, 1989.

[115] S. Goldwasser, S. Micali, and R. L. Rivest. A "Paradoxical" Solution to the Signature Problem. In *Proceedings of the 25th IEEE Symposium on Foundations of Computer Science - FOCS*, pages 441–448. IEEE, 1984.

[116] S. Goldwasser, S. Micali, and R. L. Rivest. A Digital Signature Scheme Secure Against Adaptative Chosen-Message Attacks. *SIAM Journal of Computing*, 17 :281–308, 1988.

[117] S. Goldwasser and Y. Oren. Definitions and Properties of Zero-Knowledge Proof Systems. *Journal of Cryptology*, 7(1) :1–32, 1994.

[118] S. Goldwasser and E. Waisbard. Transformation of Digital Signature Schemes into Designated Confirmer Signature Schemes. In *Theory of Cryptography - Proceedings of TCC'04*, volume 2951 of *Lecture Notes in Computer Science*, pages 77–100. Springer-Verlag, 2004.

[119] L. C. Guillou and J.-J. Quisquater. A Practical Zero-Knowledge Protocol Fitted to Security Microprocessor Minimizing Both Trasmission and Memory. In *Advances in Cryptology - Proceedings of Eurocrypt'88*, volume 330 of *Lecture Notes in Computer Science*, pages 123–128. Springer-Verlag, 1988.

[120] S. Hada and T. Tanaka. On the Existence of 3-Round Zero-Knowledge Protocols. In *Advances in Cryptology - Proceedings of Crypto'98*, volume 1462 of *Lecture Notes in Computer Science*, pages 408–423. Springer-Verlag, 1998.

[121] R. Hartshorne. *Algebraic Geometry*. Springer-Verlag, 1977.

[122] J. Herranz and G. Sáez. Forking Lemmas in the Ring Signatures' Scenario. In *Progress in Cryptology - Indocrypt 2003*, volume 2904 of *Lecture Notes in Computer Science*, pages 266–279. Springer-Verlag, 2003.

[123] M. Hirt and K. Sako. Efficient Receipt-Free Voting Based on Homomorphic Encryption. In *Advances in Cryptology - Proceedings of Eurocrypt 2000*, volume 1807, pages 539–556. Springer-Verlag, 2000.

[124] M. Jakobsson. Blackmailing using Undeniable Signatures. In *Advances in Cryptology - Proceedings of Eurocrypt'94*, volume 950 of *Lecture Notes in Computer Science*, pages 425–427. Springer-Verlag, 1995.

[125] M. Jakobsson, K. Sako, and R. Impagliazzo. Designated Verifier Proofs and their Applications. In *Advances in Cryptology - Proceedings of Eurocrypt'96*, volume 1070 of *Lecture Notes in Computer Science*, pages 142–154. Springer-Verlag, 1996.

[126] M. Jakobsson and C. P. Schnorr. Efficient Oblivious Proofs of Correct Exponentiation. In *Proceedings of CMS'99*, volume 152 of *International Federation for Information Processing (IFIP)*, pages 71–86. Academic Publishers, 1999.

[127] A. Joux. A One Round Protocol for Tripartite Diffie–Hellman. In *Proceedings of the Algorithmic Number Theory Symposium ANTS IV*, volume 1838 of *Lecture Notes in Computer Science*, pages 385–394. Springer-Verlag, 2000.

[128] A. Joux. The Weil and Tate Pairings as Building Blocks for Public Key Cryptosystems. In *Proceedings of the Algorithmic Number Theory Symposium ANTS V*, volume 2369 of *Lecture Notes in Computer Science*, pages 20–32. Springer-Verlag, 2002.

[129] A. Joux. Collisions in SHA-0. Crypto'04 rump session, 2004.

[130] A. Joux. A One Round Protocol for Tripartite Diffie–Hellman. *Journal of Cryptology*, 17(4) :263–276, 2004.

[131] A. Joux and K. Nguyen. Separating Decision Diffie-Hellman from Computational Diffie-Hellman in Cryptographic Groups. *Journal of Cryptology*, 16(4) :239–247, 2003.

[132] B. S. Kaliski, Jr. *Elliptic Curves and Cryptography : A Pseudorandom Bit Generator and Other Tools*. PhD thesis, Massachusetts Institute of Technology, 1988.

[133] J. Katz and N. Wang. Efficiency improvements for signature schemes with tight security reductions. In *Proceedings of the 10th ACM Conference on Computer and Communications Security*, pages 155–164. ACM Press, 2003.

[134] A. Kerckhoffs. La cryptographie militaire. *Journal des Sciences Militaires*, 9 :5–38 et 161–191, 1883.

[135] A. Kiayias, Y. Tsiounis, and M. Yung. Traceable Signatures. In *Advances in Cryptology - Proceedings of Eurocrypt'04*, volume 3027 of *Lecture Notes in Computer Science*, pages 571–589. Springer-Verlag, 2004.

[136] J. Kilian and P. Rogaway. How to Protect DES Against Exhaustive Key Search. In *Advances in Cryptology - Proceedings of Crypto'96*, volume 1109 of *Lecture Notes in Computer Science*, pages 252–267. Springer-Verlag, 1996.

[137] S. J. Kim, S. J. Park, and D. H. Won. A Nominative Signature. In *Conference on Information Security and Cryptology - Proceedings of CISC'94*, volume 4, 1994.

[138] S. J. Kim, S. J. Park, and D. H. Won. Zero-knowledge nominative signatures. In *Proceedings of PragoCrypt'96*, pages 380–392. Czech Technical University Publishing House, 1996,.

[139] N. Koblitz. Elliptic Curve Cryptosystems. *Mathematics of Computation*, 48 :203–209, 1987.

[140] N. Koblitz. Hyperelliptic Cryptosystems. *Journal of Cryptology*, 1(3) :139–150, 1989.

[141] H. Krawczyk and T. Rabin. Chameleon Hashing and Signatures. In *Proceedings of NDSS 2000*, pages 143–154, 2000.

[142] F. Laguillaumie, P. Paillier, and D. Vergnaud. Universally Convertible Directed Signatures from Bilinear Maps, 2005. Soumis.

[143] F. Laguillaumie, J. Traoré, and D. Vergnaud. Universal Forgery on Sekhar's Signature Scheme with Message Recovery. *International Journal of Computer Mathematics*, 81(12) :1493–1495, December 2004.

[144] F. Laguillaumie and D. Vergnaud. Multi-Designated Verifiers Signature Scheme. In *6th International Conference ICICS 2004*, volume 3269 of *Lecture Notes in Computer Science*, pages 495–507. Springer-Verlag, 2004.

[145] F. Laguillaumie and D. Vergnaud. Designated Verifier Signatures : Anonymity and Efficient Construction from *any* Bilinear Map. In *Security in Communication Networks - Proceedings of SCN'04*, volume 3352 of *Lecture Notes in Computer Science*, pages 107–121. Springer-Verlag, 2005.

[146] F. Laguillaumie and D. Vergnaud. Multi-designated Verifiers Signatures : Anonymity without Encryption, 2005. Soumis.

[147] F. Laguillaumie and D. Vergnaud. Short Undeniable Signatures Without Random Oracles : the Missing Link, 2005. Soumis.

[148] F. Laguillaumie and D. Vergnaud. Time-Selective Convertible Undeniable Signatures. In *Topics in Cryptology - CT-RSA'05*, volume 3376 of *Lecture Notes in Computer Science*, pages 154–171. Springer-Verlag, 2005.

[149] S. Lal and M. Kumar. Some Applications of Directed Signature Scheme, 2004. available on the arXiv.org e-Print archive `http://arxiv.org/`.

[150] T. Lange. *Fast Arithmetic on Hyperelliptic Curves.* PhD thesis, Universität-Gesamthochschule Essen, 2001.

[151] A. K. Lenstra, H. W. Lenstra, Jr., M. S. Manasse, and J. M. Pollard. The Number Field Sieve. In *Procedings of the 22nd annual ACM Symposium on Theory of Computing*, pages 564–572. ACM Press, 1990.

[152] A. K. Lenstra and Lenstra, Jr., H. W., editors. *The Development of the Number Field Sieve*, volume 1554 of *Lecture Notes in Mathematics*. Springer Verlag, 1993.

[153] H. W. Lenstra, Jr. Factoring Integers with Elliptic Curves. *Annals of Mathematics*, 1987.

[154] H. W. Lenstra, Jr. and C. Pomerance. A Rigorous Time Bound for Factoring Integers. *Journal of the American Mathematical Society*, 5(3) :483–516, 1992.

[155] B. Libert and J.-J. Quisquater. Identity Based Undeniable Signatures. In *Topics in Cryptology - Proceedings of CT-RSA'04*, volume 2964 of *Lecture Notes in Computer Science*, pages 112–125. Springer-Verlag, 2004.

[156] B. Libert and J.-J. Quisquater. Improved Signcryption from q-Diffie-Hellman Problems. In *Security in Communication Networks - Proceedings of SCN'04*, volume 3352 of *Lecture Notes in Computer Science*, pages 220–234. Springer-Verlag, 2005.

[157] C. H. Lim and P. J. Lee. Modified Maurer-Yacobi's Scheme and its Applications. In *Advances in Cryptology - Proceedings of Auscrypt'92*, volume 718 of *Lecture Notes in Computer Sience*, pages 308–323. Springer-Verlag, 1993.

[158] C. H. Lim and P. J. Lee. Directed Signatures and Application to Threshold Cryptosystems. In *Security Protocols Workshop*, volume 1189 of *Lecture Notes in Computer Science*, pages 131–138. Springer-Verlag, 1996.

[159] G. Martinet. *Contribution à une théorie de la sécurité des implémentations cryptographiques.* PhD thesis, Université de Paris 7 - École Normale Supérieure, 2004.

[160] U. Maurer and S. Wolf. Lower Bounds on Generic Algorithms in Groups. In *Advances in Cryptology - Proceedings of Eurocrypt'98*, volume 1403 of *Lecture Notes in Computer Science*, pages 72–84. Springer-Verlag, 1998.

[161] U. Maurer and S. Wolf. The Relationship Between Breaking the Diffie-Hellman Protocol and Computing Discrete Logarithms. *SIAM Journal on Computing*, 28(5) :1689–1721, 1999.

[162] U. Maurer and S. Wolf. The Diffie-Hellman protocol. *Designs, Codes, and Cryptography - Special Issue Public Key Cryptography*, 19(2/3) :147–171, 2000.

[163] A. May. *New RSA Vulnerabilities Using Lattice Reduction Methods*. PhD thesis, Universität Paderborn, 2003.

[164] A. May. Computing the RSA Secret Key is Deterministic Polynomial Time Equivalent to Factoring. In *Advances in Cryptology - Proceedings of Crypto'04*, volume 3152 of *Lecture Notes in Computer Science*, pages 213–219. Springer-Verlag, 2004.

[165] A. Menezes, T. Okamoto, and S. Vanstone. Reducing Elliptic Curve Logarithms to Logarithms in a Finite Field. In *Proceedings of the 22nd Annual ACM Symposium on the Theory of Computing*, pages 80–89. ACM Press, 1991.

[166] A. Menezes, T. Okamoto, and S. Vanstone. Reducing Elliptic Curve Logarithms to Logarithms in a Finite Field. *IEEE Transactions on Information Theory*, 39 :1639–1646, 1993.

[167] A.J. Menezes. *Elliptic Curve Public Key Cryptosystems*. Kluwer Academic Publishers, 1993.

[168] M. Michels, H. Petersen, and P. Horster. Breaking and Repairing a Convertible Undeniable Signature Scheme. In *Proceedings of the 3rd ACM Conference on Computer and Communications Security*, pages 148–152. ACM Press, 1996.

[169] M. Michels and M. Stadler. Efficient Convertible Undeniable Signature Schemes. In *Proceedings of the Fourth Annual Workshop on Selected Areas in Cryptography SAC'97*, pages 231–244, 1997.

[170] M. Michels and M. Stadler. Generic Constructions for Secure and Efficient Confirmer Signature Schemes. In *Advances in Cryptology - Proceedings of Eurocrypt'98*, volume 1403 of *Lecture Notes in Computer Science*, pages 406–421. Springer-Verlag, 1998.

[171] V. S. Miller. Short Programs for Functions on Curves. Unpublished Manuscript `http://crypto.stanford.edu/miller/miller.ps`, 1986.

[172] V. S. Miller. Use of Elliptic Curves in Cryptography. In *Advances in Cryptology - Proceedings of Crypto'85*, volume 218 of *Lecture Notes in Computer Science*, pages 417–426. Springer-Verlag, 1986.

[173] V. S. Miller. The Weil Pairing, and Its Efficient Calculation. *Journal of Cryptology*, 17(4) :235–261, 2004.

[174] S. Mitsunari, R. Sakai, and M. Kasahara. A New Traitor Tracing. *IEICE Trans. Fundamentals*, E85-A(2), 2002.

[175] A. Miyaji, M. Nakabayashi, and S. Takano. New Explicit Conditions of Elliptic Curve Traces for FR-Reduction. *IEICE Transactions on Fundamentals*, E84-A(5) :1234–1243, 2001.

[176] J. Monnerat and S. Vaudenay. Undeniable Signatures Based on Characters : How to Sign with One Bit. In *Public Key Cryptography - PKC 2004*, volume 2947 of *Lecture Notes in Computer Science*, pages 69–85. Springer-Verlag, 2004.

[177] D. Naccache, D. Pointcheval, and J. Stern. Twin Signatures : an Alternative to the Hash-and-Sign Paradigm. In *Proceedings of the 8th ACM Conference on Computer and Communications Security*, pages 20–27. ACM Press, 2001.

[178] D. Naccache and J. Stern. Signing on a Postcard. In *Proceedings of the 4th International Conference Financial Cryptography - FC 2000*, volume 1962 of *Lecture Notes in Computer Science*, pages 121–135. Springer-Verlag, 2001.

[179] M. Naor and M. Yung. Universal One-Way Hash Functions and their Cryptographic Applications. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, pages 33–43. ACM Press, 1989.

[180] M. Naor and M. Yung. Public-key Cryptosystems Provably Secure against Chosen Ciphertext Attacks. In *Proceedings of the 22rd ACM Symposium on Theory of Computing STOC'90*, pages 427–437. ACM Press, 1990.

[181] V.I. Nechaev. Complexity of a Deterministic Algorithm for the Discrete Logarithm. *Mathematical Notes*, 55(2) :165–172, 1994.

[182] National Institute of Standards and Technology (NIST). Secure hash standard (shs). FIPS Publication 180, 1993.

[183] National Institute of Standards and Technology (NIST). Announcement of weakness in the secure hash standard, 1994.

[184] W. Ogata, K. Kurosawa, and S.-H. Heng. The Security of the FDH Variant of Chaum's Undeniable Signature Scheme. In *Public Key Cryptography - PKC 2005*, Lecture Notes in Computer Science. Springer-Verlag, 2005. To appear.

[185] K. Ohta and T. Okamoto. On Concrete Security Treatment of Signatures Derived from Identification. In *Advances in Cryptology - Proceedings of Crypto'98*, volume 1462 of *Lecture Notes in Computer Science*, pages 354–369. Springer-Verlag, 1998.

[186] T. Okamoto. Designated Confirmer Signatures and Public Key Encryption are Equivalent. In *Advances in Cryptology - Proceedings of Crypto'94*, volume 839 of *Lecture Notes in Computer Science*, pages 61–74. Springer-Verlag, 1994.

[187] T. Okamoto and K. Ohta. How to Utilize the Randomness of Zero-Knowledge Proofs. In *Advances in Cryptology - Proceedings of Crypto'90*, volume 537 of *Lecture Notes in Computer Science*, pages 456–475. Springer-Verlag, 1991.

[188] T. Okamoto and D. Pointcheval. The Gap-Problems : a New Class of Problems for the Security of Cryptographic Schemes. In *Public Key Cryptography - PKC 2001*, volume 1992 of *Lecture Notes in Computer Science*, pages 104–118. Springer-Verlag, 2001.

[189] T. Okamoto and D. Pointcheval. REACT : Rapid Enhanced-security Asymmetric Cryptosystem Transform. In *Topics in Cryptology - Proceedings of CT-RSA'01*, volume 2020 of *Lecture Notes in Computer Science*, pages 159–175. Springer-Verlag, 2001.

[190] P. Paillier. *Cryptographie à clé publique basée sur la résiduosité de degré composite*. PhD thesis, École Nationale Supérieure des Télécommunications, 1999.

[191] S. J. Park, T. H. Kim, Y. A. An, and D. H. Won. A Provably Entrusted Undeniable Signature. In *Proceedings of the IEEE Singapore International Conference on Network/International Conference on Information Engineering - IEEE SICON/ICIE'95*, pages 644–648. IEEE, 1995.

[192] T. P. Pedersen. Distributed Provers with Applications to Undeniable Signatures. In *Advances in Cryptology - Proceedings of Eurocrypt'91*, volume 547 of *Lecture Notes in Computer Science*, pages 221–242. Springer-Verlag, 1991.

[193] S. C. Pohlig and M. E. Hellman. An Improved Algorithm for Computing Logarithms over GF(p) and its Cryptographic Significance. *IEEE Transactions on Information Theory*, 24 :106–110, 1978.

[194] D. Pointcheval. *Les preuves de connaissances et leurs preuves de sécurité*. PhD thesis, Université de Caen, 1996.

[195] D. Pointcheval. Self-Scrambling Anonymizers. In *Proceedings of the 4th International Conference Financial Cryptography - FC 2000*, volume 1962 of *Lecture Notes in Computer Science*, pages 259–275. Springer-Verlag, 2001.

[196] D. Pointcheval. Practical Security in Public-Key Cryptography. In *Proceedings of the 4th International Conference on Information Security and Cryptology (ICISC '01)*, volume 2288 of *Lecture Notes in Computer Science*, pages 1–17. Springer-Verlag, 2002.

[197] D. Pointcheval and J. Stern. Provably Secure Blind Signature Schemes. In *Advances in Cryptology - Proceedings of Asiacrypt'96*, volume 1163 of *Lecture Notes in Computer Science*, pages 252–265. Springer-Verlag, 1996.

[198] D. Pointcheval and J. Stern. Security Proofs for Signature Schemes. In *Advances in Cryptology - Proceedings of Eurocrypt'96*, volume 1070 of *Lecture Notes in Computer Science*, pages 387–398. Springer-Verlag, 1996.

[199] D. Pointcheval and J. Stern. Security Arguments for Digital Signatures and Blind Signatures. *Journal of Cryptology*, 13(3) :361–396, 2000. Full version of [197, 198].

[200] J. M. Pollard. A Monte Carlo Method for Factorization. *Nordisk Tidskrift for Informationsbehandlung (BIT)*, 15 :331–334, 1975.

[201] G. Poupard. *Authentification d'entités, de messages et de clés cryptographiques : théorie et pratique*. PhD thesis, École polytechnique, 2000.

[202] G. Poupard and J. Stern. Security Analysis of a Practical "on the fly" Authentication and Signature Generation. In *Advances in Cryptology - Proceedings of Eurocrypt'98*, volume 1403 of *Lecture Notes in Computer Science*, pages 422–436. Springer-Verlag, 1998.

[203] J.-J. Quisquater and L. Guillou. The New Guillou-Quisquater Scheme. In *Proceedings of the RSA 2000 conference*, 2000.

[204] C. W. Rackoff and D. R. Simon. Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack. In *Advances in Cryptology - Proceedings of Crypto'91*, volume 576 of *Lecture Notes in Computer Science*, pages 433–444. Springer-Verlag, 1992.

[205] R. L. Rivest. The MD4 Message Digest Algorithm, April 1992. RFC 1320, The Internet Engineering Task Force.

[206] R. L. Rivest. The MD5 Message Digest Algorithm, April 1992. RFC 1321, The Internet Engineering Task Force.

[207] R. L. Rivest. Chaffing and Winnowing : Confidentiality without Encryption. CryptoBytes (RSA Laboratories), 4 (1) : 12–17, 1998.

[208] R. L. Rivest, A. Shamir, and L. M. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21(2) :120–126, 1978.

[209] R. L. Rivest, A. Shamir, and Y. Tauman. How to Leak a Secret. In *Advances in Cryptology - Proceedings of Asiacrypt'01*, volume 2248 of *Lecture Notes in Computer Science*, pages 552–565. Springer-Verlag, 2001.

[210] J. Rompel. One-way functions are necessary and sufficient for secure signatures. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 387–394. ACM Press, 1990.

[211] S. Saeednia, S. Kremer, and O. Markowitch. An Efficient Strong Designated Verifier Signature Scheme. In *Information Security and Cryptology - ICISC 2003*, volume 2836 of *Lecture Notes in Computer Science*, pages 40–54. Springer-Verlag, 2003.

[212] A. De Santis, S. Micali, and G. Persiano. Non-Interactive Zero-Knowledge Proof Systems. In *Advances in Cryptology - Proceedings of Crypto'87*, volume 293 of *Lecture Notes in Computer Science*, pages 52–72. Springer-Verlag, 1987.

[213] T. Satoh. On p-adic point counting algorithms for elliptic curves over finite fields. In *Proceedings of the Algorithmic Number Theory Symposium ANTS V*, volume 2369 of *Lecture Notes in Computer Science*, pages 43–66. Springer-Verlag, 2002.

[214] T. Satoh and K. Araki. Fermat Quotients and the Polynomial Time Discrete Log Algorithm for Anomalous Elliptic Curves. *Commentarii Math. Univ. St. Pauli*, 47 :81–92, 1998.

[215] C. P. Schnorr. Efficient Identification and Signatures for Smart Cards. In *Advances in Cryptology - Proceedings of Crypto'89*, volume 435 of *Lecture Notes in Computer Science*, pages 239–251. Springer-Verlag, 1990.

[216] C.-P. Schnorr. Efficient Signature Generation by Smart Cards. *Journal of Cryptology*, 4(3) :161–174, 1991.

[217] R. Schoof. Counting points on elliptic curves over finite fields. *Journal de théorie des nombres de Bordeaux*, pages 219–254, 1995.

[218] M. Scott and P. S. L. M. Barreto. Compressed Pairings. In *Advances in Cryptology - Proceedings of Crypto'04*, volume 3152 of *Lecture Notes in Computer Science*, pages 140–156. Springer-Verlag, 2004.

[219] M. Scott and P. S. L. M. Barreto. Generating More MNT Elliptic Curves. *Designs, Codes and Cryptography*, To appear.

[220] M.R. Sekhar. Signature Scheme with Message Recovery and its Application. *International Journal of Computer Mathematics*, 81(3) :285–289, mars 2004.

[221] I. A. Semaev. Evaluation of Discrete Logarithms in a Group of p-Torsion Points of an Elliptic Curve in Characteristic p. *Journal of Cryptology*, 11 :141–145, 1998.

[222] R. Senderek. Personal Communication with Ronald L. Rivest, 2000. `http://theory.lcs.mit.edu/~rivest/chaffing-remarks.txt`.

[223] A. Shamir. Identity-Based Cryptosystems and Signature Schemes. In *Advances in Cryptology - Proceedings of Crypto'84*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer-Verlag, 1985.

[224] C. E. Shannon. A Mathematical Theory of Communication. *Bell System Technical Journal*, 27 :379–423 and 623–656, 1948.

[225] C. E. Shannon. Communication Theory of Secrecy Systems. *Bell System Technical Journal*, 28 :656–715, 1949.

[226] R. Shipsey. *Elliptic Divisibility Sequences*. PhD thesis, University of London, 2000.

[227] V. Shoup. Lower Bounds for Discrete Logarithms and Related Problems. In *Advances in Cryptology - Proceedings of Eurocrypt'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–266. Springer-Verlag, 1997.

[228] V. Shoup. OAEP Reconsidered. *Journal of Cryptology*, 15(4) :223–249, September 2002.

[229] J. H. Silverman. *The Arithmetic of Elliptic Curves*. Springer-Verlag, 1986.

[230] N. P. Smart. The Discrete Logarithm Problem on Elliptic Curves of Trace One. *Journal of Cryptology*, 12 :193–196, 1999.

[231] R. Steinfeld, L. Bull, H. Wang, and J. Pieprzyk. Universal Designated Verifier Signatures. In *Advances in Cryptology - Proceedings of Asiacrypt'03*, volume 2894 of *Lecture Notes in Computer Science*, pages 523–542. Springer-Verlag, 2003.

[232] R. Steinfeld, H. Wang, and J. Pieprzyk. Efficient Extension of Standard Schnorr/RSA Signatures into Universal Designated-Verifier Signatures. In *Public Key Cryptography - PKC 2004*, volume 2947 of *Lecture Notes in Computer Science*, pages 86–100. Springer, 2004.

[233] J. Stern. *La science du secret.* Odile Jacob, 1998.

[234] W. Susilo, F. Zhang, and Y. Mu. Identity-based Strong Designated Verifier Signatures Schemes. In *Proceedings of ACISP'04*, volume 3108 of *Lecture Notes in Computer Science*, pages 313–324. Springer-Verlag, 2004.

[235] T. Takagi. *New public-key cryptosystems with fast decryption.* PhD thesis, TU Darmstadt, 2001.

[236] E. Thomé. Computation of discrete logarithms in $\mathbb{F}_{2^{607}}$. In *Advances in Cryptology - Proceedings of Asiacrypt'01*, volume 2248 of *Lecture Notes in Computer Science*, pages 107–124. Springer-Verlag, 2001.

[237] A. Turing. On Computable Numbers with an Application to the *Entscheidungsproblem*. In *Proceedings of the London Mathematical Society*, volume 42 of *2*, pages 230–265, 1937. Correction, Vol. 43, 544–546.

[238] V. D. Tô, R. Safavi-Naini, and F. Zhang. New Traitor Tracing Schemes Using Bilinear Map. In *2003 ACM Workshop on Digital Rights Management (DRM 2003)*, pages 67–76. ACM Press, 2003.

[239] B. Vallée. Generation of Elements with Small Modular Squares and Provably Fast Integer Factoring Algorithm. *Mathematics of Computations*, 56 :823–849, 1991.

[240] P.C. van Oorschot and M.J. Wiener. Parallel Collision Search with Cryptanalytic Applications. *Journal of Cryptology*, 12(1) :1–28, 1999.

[241] G. Wang. An Attack on Not-interactive Designated Verifier Proofs for Undeniable Signatures. Cryptology ePrint Archive, Report 2003/243, 2003. `http://eprint.iacr.org/`.

[242] X. Wang, D. Feng, X. Lai, and H. Yu. Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD. Cryptology ePrint Archive, Report 2004/199 - Crypto'04 rump session, 2004. `http://eprint.iacr.org/`.

[243] X. Wang, Y. L. Yin, and H. Yu. Collision Search Attack on SHA1. `http://theory.csail.mit.edu/~yiqun/`, 2005.

[244] A. Weil. Sur les fonctions algébriques à corps de constantes finis. *Comptes rendus de l'Académie des sciences*, 210 :592–594, 1940.

[245] A. C.-C. Yao. Protocols for Secure Computations. In *Proceedings of the 23rd IEEE Symposium on Foundations of Computer Science - FOCS*, pages 160–164. IEEE, 1982.

[246] F. Zhang and K. Kim. A Universal Forgery on Araki *et al.*'s Convertible Limited Verifier Signature Scheme. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E86-A(2), 2003.

[247] F. ZHANG, R. SAFAVI-NAINI, and W. SUSILO. Efficient Verifiably Encrypted Signature and Partially Blind Signature from Bilinear Pairings. In *Progress in Cryptology - Indocrypt 2003*, volume 2904, pages 191–204. Springer-Verlag, 2003.

[248] F. ZHANG, R. SAFAVI-NAINI, and W. SUSILO. An Efficient Signature Scheme from Bilinear Pairings and Its Application. In *Public Key Cryptography - PKC 2004*, volume 2947 of *Lecture Notes in Computer Science*, pages 277–290. Springer-Verlag, 2004.

Résumé de la thèse.— Pour répondre à la demande croissante et variée de sécurisation des systèmes de communication, la cryptologie doit apporter des solutions adaptées, flexibles et efficaces. Parmi les besoins engendrés par ce phénomène, celui de l'authentification est fondamental.

Dans cette thèse, nous analysons des primitives cryptographiques liées à l'authentification de l'origine des données. Ce sont des variantes de la signature numérique classique qui ont comme particularités de ne pouvoir être vérifiées que sous contrôle d'une entité spéciale, et de posséder des propriétés d'anonymat.

Dans un premier temps, nous nous intéressons au concept de signatures indéniables que nous enrichissons en ajoutant une conversion temporelle des signatures. Nous menons une analyse précise de la sécurité de nouveaux schémas dans le modèle de l'oracle aléatoire, puis nous donnons un schéma de signatures indéniables dont la sécurité peut être prouvée dans le modèle standard. Par la suite, nous étudions une variante de ces signatures indéniables, appelées signatures dirigées, et proposons un nouveau schéma dont la sécurité est étudiée dans le modèle de l'oracle aléatoire. Finalement nous développons le concept de signatures à vérificateur désigné en étendant sa définition pour autoriser la présence de plusieurs vérificateurs, et proposons plusieurs schémas. Nous focalisons également notre étude sur une propriété d'anonymat du signataire.

Les signatures présentées dans cette thèse sont basées sur les couplages de type Weil et Tate, récemment introduits en cryptographie. Ils apportent un degré de liberté supplémentaire pour concevoir des cryptosystèmes, et induisent les variantes du problème Diffie-Hellman sur lesquelles repose la sécurité de nos schémas. Nous introduisons en particulier l'"astuce xyz" et le problème "xyz-DDH" grâce à de simples observations qui nous permettent d'obtenir des compromis entre authenticité et anonymat.

English title.— On the security of pairing-based signatures with controlled verification.

English abstract.— To answer to the increasing needs to secure communication systems, cryptology must provide specific, flexible and efficient solutions. Crucial is the need for authentication.

In this thesis, we analyse cryptographic primitives related to data authentication. They differ from traditional digital signatures as they can only be verified under the control of a specific entity and they provide some anonymity properties.

We first deal with undeniable signatures and introduce a new property in their definition, namely the time-selective conversion. We analyze the security of several new schemes in the random oracle model, and we propose simple undeniable signatures whose security can be proven in the standard model. We also study a logical extension of undeniable signatures, called directed signatures and we propose a new scheme and its security in the random oracle model. Finally we develop the concept of designated verifier signatures and provide a formal definition of multi-designated verifiers signatures with a rigorous treatment of the security model for such an extension. We also study an anonymity property for the signer.

The signatures presented in this thesis are based on the Weil and Tate pairings, recently introduced in cryptology. These bilinear applications allow a degree of freedom to design cryptosystems, and imply the bilinear variants of the classical Diffie-Hellman problem upon which rely our new schemes. In particular we introduce from simple observations the so-called "xyz-trick" and the "xyz-DDH" problem which make it possible to design cryptographic protocols achieving a trade-off between authenticity and privacy.

Discipline.— Mathématiques et leurs applications

Mots–clé.— Cryptographie à clé publique, authentification, signatures électroniques

---

France Télécom Recherche et Développement, 42 rue des Coutures, BP 6243, 14066 Caen cedex 04
Laboratoire de Mathématiques Nicolas Oresme, CNRS UMR 3169, Université de Caen, 14032 Caen